# МИНОБРНАУКИ РОССИИ

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

# «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

**Кафедра информационных систем**

# ОТЧЕТ

**по практической работе №3**

# по дисциплине «Программирование» Тема:  'Текстовые строки как макссив**"**

|  |  |
| --- | --- |
| Студентка гр. 3372 | Козина П.С. |
| Преподаватель | Глущенко А. Г. |

Санкт-Петербург 2023

# Цель работы.

Разработать программу, позволяющая выполнять различные операции с массивами, включая сортировку и поиск.

# Основные теоретические положения.

Массив – структура данных, хранящая набор значений, идентифицируемых по индексу или набору индексов, принимающих целые значения из некоторого заданного непрерывного диапазона. Одномерный массив можно рассматривать как реализацию абстрактного типа данных. Подробная информация о массиве представлена на рисунке 1.



Рисунок 1 – Одномерный массив

Сортировка массива – это расположение элементов массива в некотором заданном порядке (по возрастанию или убыванию). Например, на рисунке 2 представлена сортировка пузырьком.

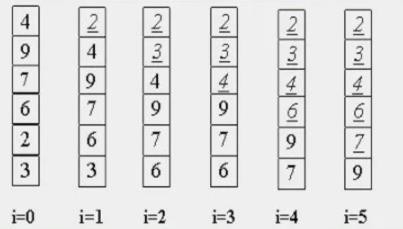


Рисунок 2 – Сортировка пузырьком В данной работе представлены следующие сортировки:

* сортировка полным перебором (от меньше к большему) – это сортировка выполняющая полный перебор всех элементов массива сравнивая и переставляя их при удовлетворенности условий;
* «быстрая сортировка» – один из самых популярных алгоритмов, в котором используется принцип «разделяй и властвуй»;
* сортировка «слиянием» – это сортировка, где мы делится массив пополам, каждый из них сортируется слиянием и потом соединяются оба массива.

Поиск элемента в массиве дает возможность удостоверится о наличии данного элемента в части массива.

В неотсортированном массиве поиск элемента необходимо просматривать весь массив (линейный поиск).

В отсортированном массиве поиск элемента можно осуществлять, не просматривая весь массив. Например, бинарный поиск представлен на рисунке 3.

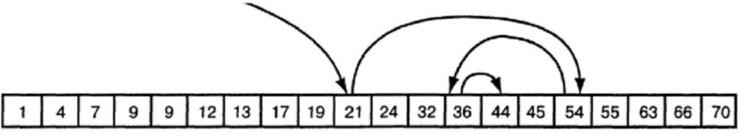


Рисунок 3 – Бинарный поиск

# Постановка задачи.

Разработать алгоритм и написать программу, которая позволяет:

1. Создать целочисленный массив размерности N = 100. Элементы массивы должны принимать случайное значение в диапазоне от -99 до 99.
2. Отсортировать заданный в пункте 1 массив […] сортировкой (от меньшего к большему). Определить время, затраченное на сортировку, используя библиотеку chrono.
3. Найти максимальный и минимальный элемент массива. Подсчитать время поиска этих элементов в отсортированном массиве и неотсортированном, используя библиотеку chrono.
4. Выводить среднее значение (если необходимо, число нужно округлить) максимального и минимального значения в отсортированном и неотсортированном. Выводить индексы всех элементов, которые равны этому значению, и их количество. Подсчитать время поиска.
5. Выводить количество элементов в отсортированном массиве, которые меньше числа a, которое инициализируется пользователем.
6. Выводить количество элементов в отсортированном массиве, которые больше числа b, которое инициализируется пользователем.
7. Выводит информацию о том, есть ли введенное пользователем число в отсортированном массиве. Реализуйте алгоритм бинарного поиска. Сравните скорость его работы с обычным перебором.
8. Менять местами элементы массива, индексы которых вводит пользователь. Выводить скорость обмена, используя библиотеку chrono.

# Выполнение работы.

Код программы представлен в приложении А.

# Выводы.

Данная практическая работа позволяет сформировать навыки и умения работы с массивами, усвоить принцип сортировки и поиска, сравнить время работы различных функций и методов.

# Ход работы

1. При запуске программы необходимо выбрать номер действия (рисунок 1).

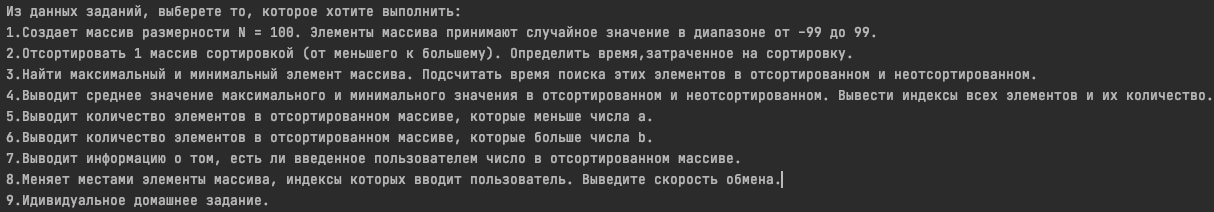


Рисунок 1 – Выбор номер поведения программы

1. При выборе пункта «1» выводится массив размерности N = 100. Элементы массива принимают случайное значение в диапазоне от -99 до 99 (рисунок 2).

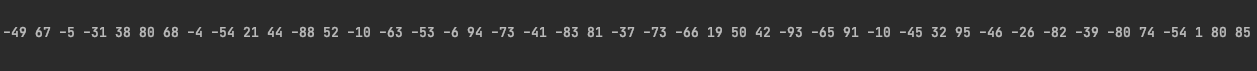


Рисунок 2 – Вывод массива

1. При выборе пункта «2» осуществляется сортировка «пузырьком» и отображение время выполнения сортировки (рисунок 3).

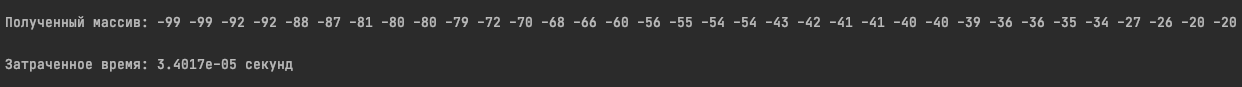


Рисунок 3 – Сортировка «пузырьком»

1. При выборе пункта «3» осуществляется поиск минимального и максимального элемента в неотсортированном и отсортированном массиве и расчет времени этого поиска (рисунок 4).

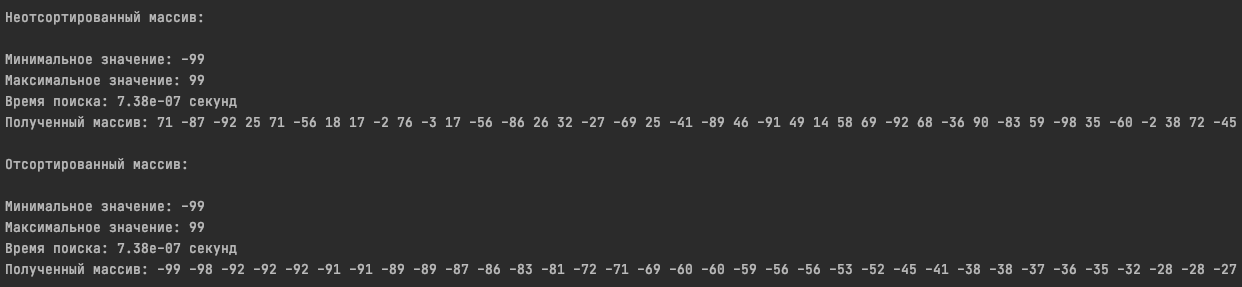


Рисунок 4 – Сортировка «слиянием»

1. При выборе пункта «4» отображается среднее минимального и максимального элемента в неотсортированном и отсортированном массиве, также выводятся индексы всех элементов и их количество и время поиска (рисунок 5)

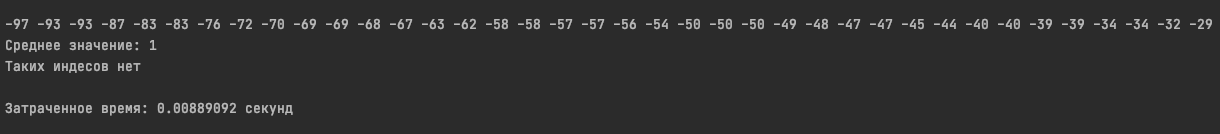


Рисунок 5 – Подсчет и поиск среднего в массиве в неотсортированном и отсортированном массиве

6. При выборе пункта «5» и «6» отображается количество элементов массива, меньше и больше введенного значения пользователя соответственно (рисунок 6,7).

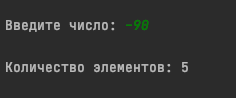


Рисунок 6 – Отображение количества элементов, меньше введённого значения

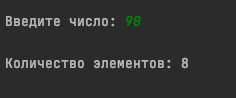


Рисунок 7 – Отображение количества элементов, больше введённого значения

1. При выборе пункта «7» выводится информацию о том, есть ли введенное пользователем число в отсортированном массиве, также сравнивает скорость работы бинарного поиска с обычным перебором (рисунок 8)

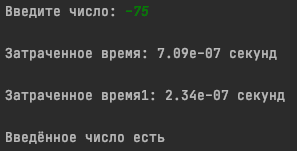


Рисунок 8 – Отображение информации о том, что есть введённое число в массиве и сравнение скорости работы бинарного поиска с обычным перебором

1. При выборе пункта «8» осуществляется смена элементов массива при помощи индексов, введенных пользователем и выводится скорость обмена (рисунок 9)

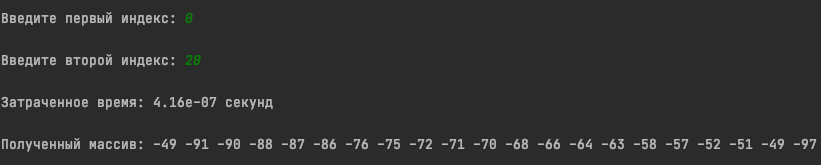


Рисунок 9 – Смена элементов

# ПРИЛОЖЕНИЕ А РАБОЧИЙ КОД

Листинг А.1 – Рабочий код программы

#include <iostream>

#include <string>

#include <vector>

#include <fstream>

#include <algorithm>

#include <vector>

using namespace std;

bool compareFunction(string a, string b) { return a < b; }

void linear(string text, string substring)

{

bool is\_substring;

for (int i = 0; i < text.length() - substring.length() + 1; i++)

{

if (substring.length() == 0)

{

cout << "Пустая подстрока\n";

return;

}

is\_substring = true;

for (int j = 0; j < substring.length(); j++) {

if (text[i + j] != substring[j]) {

is\_substring = false;

break;

}

}

if (is\_substring) {

cout << "Подстрока найдена по индексу: " << i << endl;

}

}

}

void KMP(string text, string substring)

{

int m = text.length();

int n = substring.length();

// если шаблон это пустая строка

if (n == 0)

{

cout << "Пустая подстрока\n";

return;

}

// если длина текста меньше длины шаблона

if (m < n)

{

return;

}

// next[i] сохраняет индекс следующего лучшего частичного совпадения

int\* next = new int[n + 1];

for (int i = 0; i < n + 1; i++) {

next[i] = 0;

}

for (int i = 1; i < n; i++)

{

int j = next[i];

while (j > 0 && substring[j] != substring[i]) {

j = next[j];

}

if (j > 0 || substring[j] == substring[i]) {

next[i + 1] = j + 1;

}

}

for (int i = 0, j = 0; i < m; i++)

{

if (text[i] == substring[j])

{

if (++j == n) {

cout << "Подстрока найдена по индексу: " << i - j + 1 << endl;

}

}

else if (j > 0)

{

j = next[j];

i--; // так как `i` будет увеличен на следующей итерации

}

}

}

int main()

{

setlocale(LC\_ALL, "ru");

srand(time(NULL));

string s;

int input\_choose = 0;

int task\_choose = 0;

while (true)

{

int task\_choose;

while (true)

{

cout << "\n\n\nВыберите задание, которое хотите выполнить:\n"

"1.С клавиатуры вводится последовательность, содержащая от 1 до 50 слов, в каждом из которых от 1 до 10 строчных латинских букв и цифр. Между словами произвольное количество пробелов. За последним символом стоит точка\n"

"2.Редактирование входного текста\n"

"3.Вывести на экран слова последовательности в алфавитном порядке\n"

"4.Вывести на экран все слова последовательности в две или три колонки (в зависимости от количества слов) с выравниванием слов по правой границе колонки\n"

"5.Необходимо найти все подстроки, которую введёт пользователь в имеющейся строке. Реализуйте алгоритм Линейный поиск и КМП\n";

cin >> task\_choose;

if (task\_choose == -1)

{

break;

}

if (task\_choose != 1 && task\_choose != 2 && task\_choose != 3 && task\_choose != 4 && task\_choose != 5)

{

cout << "Введите корректное значение! \n";

continue;

}

switch (task\_choose)

{

case 1:

{

cout << "\n\n\nВыберите способ ввода: \n"

"1.Ручной ввод\n"

"2.Ввод содержимого text.txt\n";

cin >> input\_choose;

cout << "\n";

if (input\_choose == -1)

{

return 0;

}

if (input\_choose != 1 && input\_choose != 2)

{

cout << "Введите корректное значение! \n";

continue;

}

else {

s = "";

switch (input\_choose)

{

case 1:

{

cout << "Введите список слов:\n";

getline(cin, s);

getline(cin, s);

cout << "\n";

break;

}

case 2:

{

ifstream in("text.txt");

string line;

if (in.is\_open())

{

while (getline(in, line))

{

s += line;

}

}

in.close();

break;

}

}

vector <string> words;

string cur\_word = "";

bool extra\_space = false;

for (int i = 0; i < (s.length() - 1); i++) {

if (s.at(i) == ' ') {

if (!extra\_space) {

words.push\_back(cur\_word);

cur\_word = "";

}

extra\_space = true;

continue;

}

cur\_word += s.at(i);

extra\_space = false;

}

words.push\_back(cur\_word);

cout << "Введенный список слов:\n\n";

for (int i = 0; i < words.size(); i++) {

cout << i + 1 << ": " << words[i] << "\n";

}

break;

}

case 2:

{

cout << "Введите текст: ";

getline(cin, s);

getline(cin, s);

//Удаление лишних пробелов

bool previousIsSpace = false;

for (auto it = s.begin(); it != s.end();) {

if (std::isspace(\*it)) {

if (previousIsSpace) {

it = s.erase(it);

}

else {

previousIsSpace = true;

++it;

}

}

else {

previousIsSpace = false;

++it;

}

}

// Удаление лишних знаков препинания

for (auto it = s.begin(); it != s.end();) {

if (std::ispunct(\*it)) {

auto next = std::next(it);

while (next != s.end() && std::ispunct(\*next)) {

next = s.erase(next);

}

++it;

}

else {

++it;

}

}

// Исправление регистра букв

for (auto& ch : s) {

if (std::isalpha(ch)) {

if (std::islower(ch)) {

ch = std::tolower(ch);

}

else {

ch = std::tolower(ch);

}

}

}

std::cout << "Отредактированный текст: " << s;

break;

}

case 3:

{

cout << "\n\n\nВыберите способ ввода: \n"

"1.Ручной ввод\n"

"2.Ввод содержимого text.txt\n";

cin >> input\_choose;

cout << "\n";

if (input\_choose == -1)

{

return 0;

}

if (input\_choose != 1 && input\_choose != 2)

{

cout << "Введите корректное значение! \n";

continue;

}

else

switch (input\_choose)

{

case 1:

{

cout << "Введите список слов:\n";

getline(cin, s);

getline(cin, s);

cout << "\n";

break;

}

case 2:

{

ifstream in("text.txt");

string line;

if (in.is\_open())

{

while (getline(in, line))

{

s += line;

}

}

in.close();

break;

}

}

vector <string> words;

string cur\_word = "";

bool extra\_space = false;

for (int i = 0; i < (s.length() - 1); i++) {

if (s.at(i) == ' ') {

if (!extra\_space) {

words.push\_back(cur\_word);

cur\_word = "";

}

extra\_space = true;

continue;

}

cur\_word += s.at(i);

extra\_space = false;

}

words.push\_back(cur\_word);

cout << "Введенный список слов:\n\n";

for (int i = 0; i < words.size(); i++) {

cout << i + 1 << ": " << words[i] << "\n";

}

sort(words.begin(), words.end(), compareFunction);

cout << "\nОтсортированный список слов:\n\n";

for (int i = 0; i < words.size(); i++) {

cout << i + 1 << ": " << words[i] << "\n";

}

break;

}

case 4:

{

cout << "\n\n\nВыберите способ ввода: \n"

"1.Ручной ввод\n"

"2.Ввод содержимого text.txt\n";

cin >> input\_choose;

cout << "\n";

if (input\_choose == -1)

{

return 0;

}

if (input\_choose != 1 && input\_choose != 2)

{

cout << "Введите корректное значение! \n";

continue;

}

else

switch (input\_choose)

{

case 1:

{

cout << "Введите список слов:\n";

getline(cin, s);

getline(cin, s);

cout << "\n";

break;

}

case 2:

{

ifstream in("text.txt");

string line;

if (in.is\_open())

{

while (getline(in, line))

{

s += line;

}

}

in.close();

break;

}

}

vector <string> words;

string cur\_word = "";

bool extra\_space = false;

for (int i = 0; i < (s.length() - 1); i++) {

if (s.at(i) == ' ') {

if (!extra\_space) {

words.push\_back(cur\_word);

cur\_word = "";

}

extra\_space = true;

continue;

}

cur\_word += s.at(i);

extra\_space = false;

}

words.push\_back(cur\_word);

int n\_columns = words.size() > 12 ? 3 : 2;

string temp;

cout << "Введенный список слов в виде таблицы:\n\n";

for (int i = 0; i < words.size(); i++) {

temp = string((20 - words[i].length()), ' ') + words[i];

cout << temp;

if ((i + 1) % n\_columns == 0) { cout << "\n"; }

}

break;

}

case 5:

{

cout << "Введите строку:\n";

getline(cin, s);

getline(cin, s);

cout << endl;

string substring;

cout << "Введите подстроку:\n";

getline(cin, substring);

cout << endl;

cout << "Линейный поиск\n";

linear(s, substring);

cout << "КМП\n";

KMP(s, substring);

break;

}

}

}

}

break;

}

return 0;

}