|  |
| --- |
| A picture containing diagram  Description automatically generated |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА - Российский технологический университет»**  **РТУ МИРЭА** |

Отчет по выполнению практических заданий

по дисциплине **«Структуры и алгоритмы обработки данных»**

Выполнил студент Мусаилов А.С.

группа ИКБО-08-21

**Москва 2022**

Содержание

[Практическая работа 1 3](#_Toc102812148)

[Практическая работа 2 6](#_Toc102812149)

# Практическая работа 1

## Цель работы

* Приобретение навыков по определению одномерного массива для структуры данных задачи
* Приобретение навыков создания алгоритмов операций над одномерным массивом
* Получение навыков по реализации алгоритмов операций над массивом через аппарат функций

## Задание

* 1. Постановка задачи: Разработать программу для выполнения операций на статическом, динамическом массивах и векторе
  2. Модель решения поставленной задачи:
     1. Пользователю предлагается выбрать режим работы (статический, динамический массив или вектор). Ввод элементов массива, далее выбор функции согласно варианту: Найти индекс элемента массива, значение которого содержит цифру 0; Вставить новый элемент в массив после элемента, значение которого не содержит цифру 0; Удалить элементы массива, значение которого содержит цифру 0.

## Реализация

bool zero\_in\_num(int x) {

while (x > 0) {

if (x % 10 == 0) {

return true;

}

x = x / 10;

}

return false;

}

int func1static(int\* array, int n)

{

int i = 0;

while (i < n) {

if (zero\_in\_num(array[i])) {

return i;

}

i += 1;

}

return -1;

}

Ф-я zero\_in\_num определяет, содержится ли число 0 в записи числа. Ф-я func1static возвращает индекс первого элемента, в записи которого есть 0.

int func2static(int\* array, int &n, int x)

{

if (n + 1 >= 11) {

return -2;

}

int i = 0;

while (i < n) {

if (!zero\_in\_num(array[i])) {

i++;

break;

}

i += 1;

}

if (i == 0) {

return -1;

}

for (int j = n; j > i; j--) {

array[j] = array[j - 1];

}

array[i] = x;

n++;

return 0;

}

Ф-я func2static добавляет введенный пользователем элемент после элемента, запись которого не содержит 0.

void func3static(int\* array, int& n)

{

int i = 0;

while (i < n) {

if (zero\_in\_num(array[i])) {

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

array[j] = array[j + 1];

}

n--;

}

else i++;

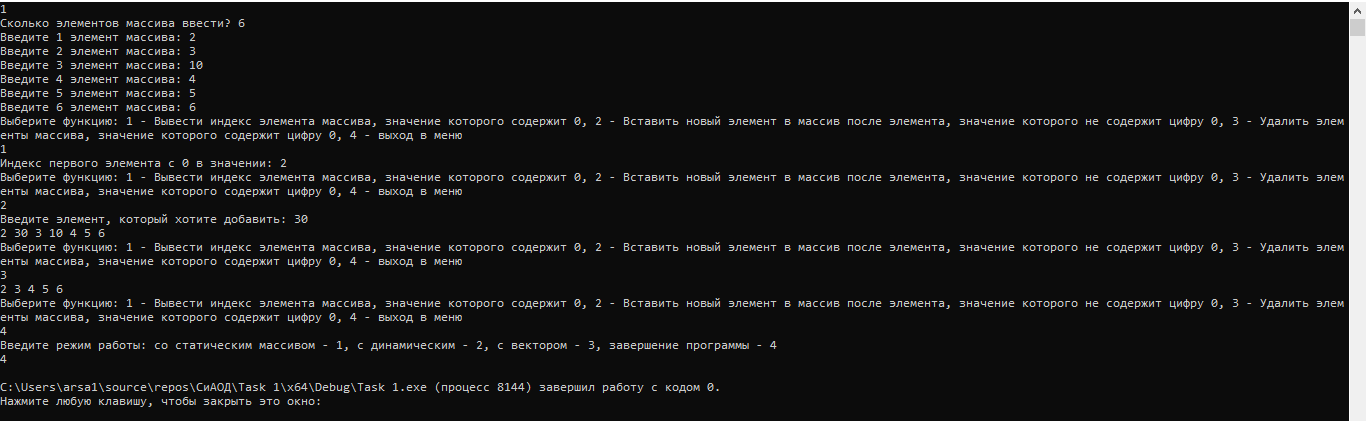
}

}

Ф-я func3static удаляет элементы, запись которых содержит 0.

Функции для других видов контейнеров (динамический массив, вектор) реализованы аналогично с учётом особенностей работы с ними.

## Вывод по заданию 1.



## Список информационных источников

1. Работа с массивами, вектором: <https://en.cppreference.com/w/>

## Общий вывод

На практике закреплены навыки работы с различными видами контейнеров в языке С++

# Практическая работа 2

## Цель работы:

* Получение навыков по определению двумерного массива для структуры данных задачи
* Получение навыков по разработке алгоритмов операций на двумерном массиве в соответствии с задачей

## Задание 1

1. Постановка задачи: разработать программу по обработке данных, представленных в задаче матрицей и реализованной в программе двумерным (многомерным) массивом, реализованным с помощью статического, динамического массивов.
2. Задание варианта: *Шахматная доска 8\*8. Вводятся координаты ферзя. Отметить клетки (вставить какое-то значение) поля доски, которые находятся под боем ферзя. Обращение к полям доски выполнять по правилам игры в шахматы*
3. Модель решения поставленной задачи:
   1. Пользователю предлагается ввести координаты доски, на которых будет стоять ферзь. Далее создаётся двумерный массив, заполненный нулями. Клетка ферзя обозначается 8, клетки под боем – 1. Шахматная доска выводится в консоль.

## Реализация

void input\_coordinates(int &row, int &col)

{

cout << "Введите номер строки, на котором будет стоять ферзь: (1-8) " << endl;

while (!(cin >> row)) {

cout << "Неверное значение" << endl;

}

cout << "Введите номер столбца, на котором будет стоять ферзь: (1-8) " << endl;

while (!(cin >> col)) {

cout << "Неверное значение" << endl;

}

}

void func\_static(int (&array)[8][8], int row, int col)

{

for (int i = 0; i <= 8; i++)

{

for (int j = 0; j <= 8; j++)

{

if (array[i][j] != 8)

{

if (i == row || j == col || (j - i == col - row) || (j + i == col + row)) array[i][j] = 1;

}

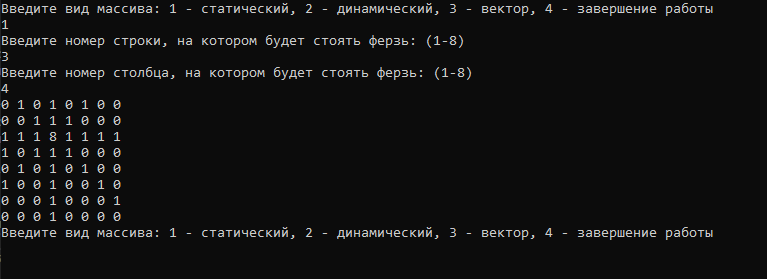
}

}

}

Ф-я input\_coordinates определяет клетку, на которой будет стоять ферзь, введенную пользователем. Ф-я func\_static заполняет шахматную доску полями под боем.

## Вывод по заданию 1



## Задание 2

1. Постановка задачи: Разработать программу по задаче варианта с реализацией данных в задаче с применением вектора библиотеки STL
2. Задание варианта: *Текст вводится как последовательность символов, заканчивающаяся точкой. Определить количество различных пар букв в этом тесте.*
3. Модель решения:
   1. Вводится строка line. Символ symb – первый элемент строки. Циклом проходимся по строке, если элемент совпадает с предыдущим, то увеличим счётчик на 1. Меняем symb на текущий элемент. Если текущий элемент 0 – выходим из цикла. Выводим в консоль результат.

## Реализация

cout << "Введите строку из символов: ";

string line;

cin >> line;

char x = line[0];

int count = 0;

for (int i = 1; i < line.size(); i++) {

if (line[i] == x) count++;

x = line[i];

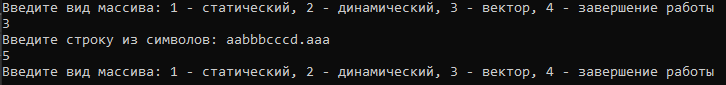
if (line[i] == '.') break;

}

cout << count << endl;

break;

## Вывод по заданию 2



## Общий вывод

На практике закреплены навыки работы с двумерными массивами.

## Список информационных источников

1. Работа с массивами, вектором, методы строк: <https://en.cppreference.com/w/>

# Практическая работа 3

## Цель работы:

* Получение навыков в разработке алгоритмов обработки текста – извлечение отдельных элементов.
* Получение навыков использования средств языка Си и С++ для реализации алгоритмов обработки текстовых данных

## Задание

1. Постановка задачи: разработать программу согласно задаче варианта, используя для представления в программе текста:
   1. Нуль терминальную строку и средства языка С для выполнения операций над этой строкой
   2. строку string, стандартной библиотеки шаблонов и возможности класса для выполнения действий со строкой
2. Задача варианта: *Дано предложение, состоящее из слов, разделенных запятой или пробелами. Слова, которые больше последнего слова, заменить на их перевертыши, а слова, которые меньше последнего, занести в массив слов и удалить из предложения.*
3. Модель решения: Пользователь выбирает тип строки (нуль-терминальная или string), происходит ввод строки, над ней выполняются действия согласно варианту, вывод итогового результата в консоль

## Реализация

string find\_last\_word(string text)

{

string word;

for (int i = 0; i < text.length(); i++) {

if ((text[i] <= 90 && text[i] >= 65) || (text[i] <= 122 && text[i] >= 97)) {

word += text[i];

}

else {

word = "";

}

}

return word;

}

Ф-я find\_last\_word возвращает последнее слово в тексте, состоящее только из букв.

void erase\_words(string& text, int length)

{

int index = 0;

string word;

int len = text.length();

int i(0);

while (i != text.length() - 1) {

if ((text[i] <= 90 && text[i] >= 65) || (text[i] <= 122 && text[i] >= 97)) {

word += text[i];

if (word.length() == 1) index = i;

i++;

}

else {

if (word.length() < length && word != "") {

text.erase(index, word.length());

word = "";

i = index;

}

else {

i++;

word = "";

}

}

}

}

Ф-я erase\_words удаляет слова, длина которых меньше длины последнего слова

void reverse\_words(string& text, int length)

{

vector <char> w;

int index(0);

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

if ((text[i] <= 90 && text[i] >= 65) || (text[i] <= 122 && text[i] >= 97)) {

w.push\_back(text[i]);

if (w.size() == 1) index = i;

}

else {

if (w.size() > length) {

for (int j = index; j < i; j++) {

text[j] = w.back();

w.resize(w.size() - 1);

}

}

w.clear();

}

}

}

Ф-я переворачивает слова, длина которых больше длины последнего слова.

char text[] = "aaa ssdsa aad";

int len = strlen(text);

char result[500]{};

int lastWordLength;

lastWordLength = find\_last\_word(text).size();

char word[500] = {};

char w[2] = {};

for (int i = 0; i < strlen(text) - lastWordLength; i++) {

if ((text[i] <= 90 && text[i] >= 65) || (text[i] <= 122 && text[i] >= 97)) {

w[0] = { text[i] };

strcat\_s(word, w);

}

else {

if (strlen(word) > lastWordLength) {

char reversed[50]{};

for (int i = strlen(word); i >= 0; i--) {

w[0] = { word[i] };

strcat\_s(result, w);

}

strcat\_s(result, " ");

}

strcpy\_s(word, "");

}

}

char lastWord[50]{};

for (int i = strlen(text) - lastWordLength; i <= strlen(text); i++) {

w[0] = { text[i] };

strcat\_s(lastWord, w);

}

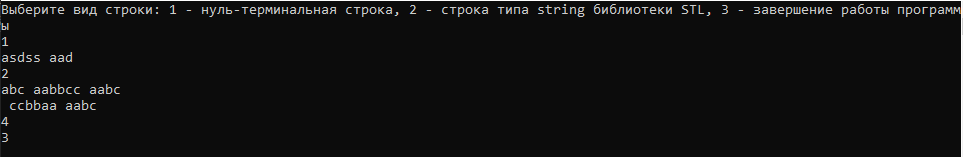
strcat\_s(result, lastWord);

cout << result << endl;

break;

Алгоритм программы для нуль-терминальной строки с учетом особенностей работы с этим типом строки.

## Вывод программы



## Общий вывод

На практике получены навыки работы с двумя видами строк.

## Список информационных источников

1. Нуль-терминальные строки, строки класса string: <https://en.cppreference.com/w/>

# Практическая работа 4

## Цель работы:

* Получение навыков по реализации многоэлементных структур данных задачи С++ .
* Реализация многоэлементных данных средствами структуры данных – таблица.

## Задание

1. Постановка задачи: Разработать набор операций для управления таблицей, созданной на основе статического и динамического массива, вектора.
2. Задание варианта:

|  |  |
| --- | --- |
| 19 | Справочник банков. Об отдельном банке хранятся данные: наименование, код банка, адрес, форма собственности (коммерческий или государственный), список услуг банка для юридических лиц (*справка:* предоставлении кредитов, автокредитов, выпуске кредитных, платежных карт, размещении денег на депозитных, текущих счетах, проведении расчетно-кассовых и других операций).  Операции   1. Заполнение записи по отдельному банку с клавиатуры (список услуг не заполнять). 2. Добавить запись по банку в таблицу. 3. Вставить в начало списка данные по предоставляемой услуге в список услуг банка. 4. Удалить запись о заданном банке. 5. Сформировать список государственных банков, предоставляемых ипотечный кредит. |

1. Модель решения: Создать структуру Bank с соответствующими полями, фунции работы с массивом структур Bank (таблицей) для разных типов контейнеров

## Реализация

struct Bank

{

string name = "";

unsigned int code;

string address = "";

string formOfOwnership = "";

bool credits;

bool cards;

};

Структура Bank содержит 6 полей. Credits – выдает ли ипотеку, cards – выдает ли дебетовые карты

void add\_bank\_static(Bank banks[], int& count\_banks)

{

if (count\_banks + 1 <= 10) {

Bank obj;

string name;

cin >> name;

obj.name = name;

int code;

cin >> code;

obj.code = code;

string address;

cin >> address;

obj.address = address;

bool credits;

cin >> credits;

obj.credits = credits;

bool cards;

cin >> cards;

obj.cards = cards;

banks[count\_banks] = obj;

count\_banks++;

}

else {

cout << "Sorry, bank list is full" << endl;

}

}

Добавление банка в таблицу.

void info\_bank\_static(Bank banks[], int code)

{

for (int i = 0; i < 10; i++) {

if (banks[i].code == code) {

string name;

cin >> name;

banks[i].name = name;

string address;

cin >> address;

banks[i].address = address;

cout << "Bank info was successfully updated" << endl;

}

}

}

Изменение информации о банке для статического массива.

void delete\_bank\_static(Bank banks[], int &count\_banks, int code)

{

for (int i = 0; i < 10; i++) {

if (banks[i].code == code) {

for (int j = i; j < 9; j++) {

banks[j] = banks[j + 1];

}

count\_banks--;

}

}

}

Удаление банка из таблицы для статического массива по коду.

void ipoteka\_static(Bank banks[])

{

for (int i = 0; i < 10; i++) {

if (banks[i].credits) {

cout << banks[i].name << " " << banks[i].address << endl;

}

}

}

Вывод банков, выдающих ипотеку.

Для других видов контейнеров функции реализованы аналогично с учетом особенностей работы с ними.

## Вывод программы



## Общий вывод

На практике получены навыки работы со структурами struct в языке С++

## Список информационных источников

1. Структуры, работа с контейнерами: <https://en.cppreference.com/w/>

# Практическая работа 5

## Цель работы:

1. Получить знания и практические навыки управления динамическим однонаправленным списком

## Задание

1. Постановка задачи: Реализуйте программу решения задачи варианта по использованию линейного однонаправленного списка.
2. Задание варианта:

|  |  |  |
| --- | --- | --- |
| 3 | char | Даны два линейных однонаправленных списка L1 и L2.   1. Разработать функцию, которая формирует список L, включив в него по одному разу элементы, значения которых входят в список L1 и не входят в список L2. 2. Разработать функцию, которая удаляет подсписок списка L1 заданный диапазоном позиций. Например, со второго три. 3. Разработать функцию, которая упорядочивает значения списка L2, располагая их в порядке возрастания. |

1. Модель решения: Создать структуру Element с полями указателя на Element next и char data для хранения данных. Разработать функции согласно варианту. Разработать дополнительные функции для ввода и вывода элементов списка.

## Реализация

struct element {

char data;

element\* next;

};

element fill\_list(int count\_elem, int mode) // Заполнение списка символьными значениями

{

element\* L = new element;

L->next = NULL;

if (mode == 1) {

for (int i = 'j'; i < 'j' + count\_elem; i++) {

element\* elem = new element;

if (elem != NULL) {

elem->data = i;

elem->next = NULL;

if (L->next == NULL) {

L->next = elem;

}

else {

element\* tmp = L;

while (tmp->next != NULL) {

tmp = tmp->next;

}

tmp->next = elem;

}

}

}

} else if (mode == 2) {

for (int i = 'z'; i > 'z' - count\_elem; i--) {

element\* elem = new element;

if (elem != NULL) {

elem->data = i;

elem->next = NULL;

if (L->next == NULL) {

L->next = elem;

}

else {

element\* tmp = L;

while (tmp->next != NULL) {

tmp = tmp->next;

}

tmp->next = elem;

}

}

}

}

return \*L;

}

Fill\_list создает элементы списка и заполняет их символами латинского алфавита в зависимости от выбранного режима работы.

bool elem\_in\_list(element\* l, char elem) // Проверка нахождения элемента в списке

{

element\* tmp = l;

while (tmp->next != NULL) {

if (tmp->data == elem) {

return true;

}

tmp = tmp->next;

}

if (tmp->data == elem)

{

return true;

}

return false;

}

bool elem\_in\_list(vector<char> v, char elem)

{

for (char i : v)

{

if (i == elem) return true;

}

return false;

}

elem\_in\_list проверяет вхождение символа в список

element make\_list(element\* L1, element\* L2)

{

element\* L = new element;

L->next = NULL;

vector <char> included\_chars; // Добавленные в искомый список символы

while (L1->next != NULL) {

L1 = L1->next;

if (!elem\_in\_list(L2, L1->data) && !elem\_in\_list(included\_chars, L1->data)) {

element \*elem = new element;

elem->data = L1->data;

elem->next = NULL;

if (L->next == NULL) L->next = elem;

else {

element\* tmp = L;

while (tmp->next != NULL) {

tmp = tmp->next;

}

tmp->next = elem;

}

included\_chars.push\_back(L1->data);

}

}

return \*L;

}

Make\_list создает однонаправленный список из элементов, входящих в L1 и не входящих в L2

element delete\_elements(element\* L1, int pos1, int pos2)

{

element to\_return = \*L1;

element\* next = new element;

int i(1); // Позиция

while (L1->next != NULL) {

next = L1->next;

if (i >= pos1 && i <= pos2) {

L1->next = next->next;

}

else {

L1 = L1->next;

}

i++;

}

return to\_return;

}

Delete\_elements удаляет элементы с индекса pos1 до индекса pos2

element sort\_list(element\* L2)

{

element to\_return = \*L2;

element \*i = L2->next;

while (L2->next != NULL) {

while (i->next != NULL) {

if (i->data < L2->data) {

char tmp = L2->data;

L2->data = i->data;

i->data = tmp;

}

i = i->next;

}

if (i->data < L2->data) {

char tmp = L2->data;

L2->data = i->data;

i->data = tmp;

}

L2 = L2->next;

if (L2->next != NULL) {

i = L2->next;

}

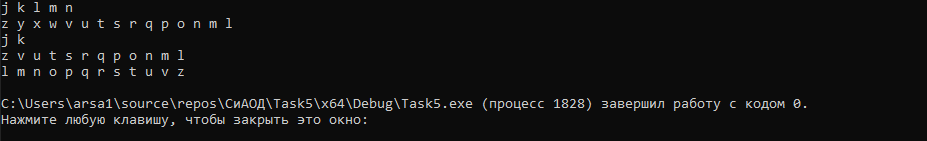
}

return to\_return;

}

Ф-я sort\_list сортирует значения списка по возрастанию.

## Вывод программы



## Общий вывод

Получены навыки работы с однонаправленным списком с помощью структур, реализованы алгоритмы функций для обработки списков.

## Список информационных источников

1. Работа с однонаправленными списками: Интернет

# Практическая работа 6

## Цель работы:

Получить знания и практические навыки по разработке и реализации рекурсивных процессов

## Задание:

1. Постановка задачи: разработать и протестировать рекурсивные функции в соответствии с задачами варианта
2. Задача варианта:

|  |  |
| --- | --- |
| 19 | 1. Дано целое десятичное число. Добавить цифру в начало числа. 2. Даны два линейных однонаправленных списка. Определить, равны ли они. Должны совпадать по длине и по значениям узлов одинаковым номером. |

1. Модель решения:
   1. Полный остаток – 0, вес младшего разряда – 1. Текущий остаток – остаток от деления числа на 10. Целочисленно делим число на 10. Если число равно 0, возвращаем число, которое нужно было добавить в начало, умноженженное на вес и на 10 + остаток. Иначе вызвать рекурсивно функцию с параметрами num, numToAdd, вес \* 10. ost
   2. Создаём структуру elem аналогично 5 работе. Создаём два списка, заполняем их элементами. Проверяем, равны ли их элементы.

## Реализация

// Добавить цифру в начало числа

int add\_digit\_at\_begin(int num, int numToAdd, int mes, int ost)

{

int cur\_ost = num % 10;

ost = cur\_ost \* mes + ost; // Полный остаток - остаток + текущий остаток умноженный на вес разряда

num = num / 10;

if (num == 0) {

return numToAdd \* (mes \* 10) + ost; // Число = цифра на вес следующего разряда + полный остаток

}

else {

add\_digit\_at\_begin(num, numToAdd, mes \* 10, ost);

}

}

element fill\_list(int count\_elem) // Заполнение списка символьными значениями

{

element\* L = new element;

L->next = NULL;

for (int i = 1; i < 1 + count\_elem; i++) {

element\* elem = new element;

if (elem != NULL) {

elem->data = i;

elem->next = NULL;

if (L->next == NULL) {

L->next = elem;

}

else {

element\* tmp = L;

while (tmp->next != NULL) {

tmp = tmp->next;

}

tmp->next = elem;

}

}

}

return \*L;

}

bool equal\_lists(element\* L1, element\* L2)

{

// Если не совпадают значения - не равны

if (L1->data != L2->data) {

return false;

}

/\* Если первый список длиннее другого или наоборот - не равны

\* Если у списков есть след. элемент - рекурсивно проверяем следующие элементы

\* Иначе равны

\*/

if (L1->next != NULL && L2->next == NULL) return false;

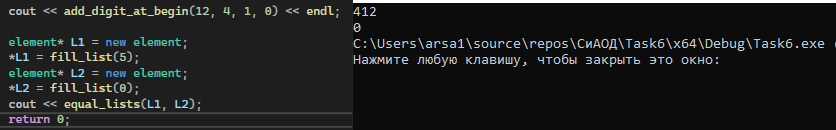
else if (L1->next == NULL && L2->next != NULL) return false;

else if (L1->next != NULL && L2->next != NULL) equal\_lists(L1->next, L2->next);

else return true;

}

## Вывод программы



## Общий вывод

На практике получены навыки работы с рекурсивными функциями.

# Список информационных источников

1. Рекурсивные функции, работа с ними, однонаправленный список: Интернет, лекции

# Практическая работа 7

## Цель работы:

* Получить знания и навыки по реализации структуры стек и очередь
* Получить умения и навыки по выполнению операций на структурах стек и очередь
* Получить знания, умения по представлению арифметических выражения в польской записи

## Задание

1. Постановка задачи:
   * Провести преобразование инфиксной записи выражения (столбец 1 таблицы вариантов) в постфиксную нотацию, расписывая процесс по шагам.
   * Представить инфиксную нотацию выражения (столбец 2 таблицы вариантов) (идентификаторы одно символьные) с расстановкой скобок, расписывая процесс по шагам.
   * Представить префиксную нотацию выражения, полученного в результате выполнения задачи 2, расписывая процесс по шагам.
   * Вычислить значение выражения, представленного в столбце 3, выполняя вычисление по образцу 4.
2. Модель решения: Создание алгоритмов перевода инфиксной записи в постфиксную, инфиксной в префиксную, вычисления постфиксной записи.

## Реализация

bool isOperator(char x)

{

return (x == '+' || x == '-' || x == '\*' || x == '/');

}

bool isOperand(char x)

{

return ((x >= 'a' && x <= 'z') || (x >= 'A' && x <= 'Z'));

}

bool higherPriority(char x, vector <char> stack)

{

return ((stack.back() == '-' || stack.back() == '+') && (x == '\*' || x == '/'));

}

Ф-я higherPriority определяет приоритет операции (true – выше, false – нет)

void retrieve(vector <char>& stack, string& postfix, char x)

{

if (x == '+' || x == '-') {

while (stack.back() != '(') {

postfix += stack.back();

stack.pop\_back();

if (stack.empty()) break;

}

stack.push\_back(x);

}

else {

while (stack.back() != '(') {

postfix += stack.back();

stack.pop\_back();

if (stack.back() == '-' || stack.back() == '+') break;

}

stack.push\_back(x);

}

}

void retrieveTo(vector <char>& stack, string& postfix)

{

while (stack.back() != '(') {

postfix += stack.back();

stack.pop\_back();

}

stack.pop\_back();

}

Ф-ии вытаскивания элемента из стека

string fromInfixToPostfix(string infix)

{

vector <char> stack;

string postfix = "";

char symbol;

while (infix.size() != 0) {

symbol = infix[0];

if (symbol == ' ') {

infix.erase(0, 1);

continue;

};

if (isOperator(symbol)) {

if (stack.size() == 0 || stack.back() == '(') {

stack.push\_back(symbol);

}

else if (higherPriority(symbol, stack)) {

stack.push\_back(symbol);

}

else {

retrieve(stack, postfix, symbol);

}

}

else if (symbol == '(') {

stack.push\_back(symbol);

}

else if (symbol == ')') {

retrieveTo(stack, postfix);

}

else {

postfix += symbol;

}

infix.erase(0, 1);

}

while (stack.size() != 0) {

postfix += stack.back();

stack.pop\_back();

}

return postfix;

}

Ф-ия перевода из инфиксной записи в постфиксную с односимвольными переменными.

string fromPostfixToInfix(string postfix) { //Получить инфиксную запись по постфиксной

vector <string> stack;

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

if (postfix[i] == ' ') continue;

if (isOperand(postfix[i])) { //Операнды - в стек

string op(1, postfix[i]);

stack.push\_back(op);

}

else {

string operator1 = stack.back();

stack.pop\_back();

string operator2 = stack.back();

stack.pop\_back();

stack.push\_back(operator2 + postfix[i] + operator1);

}

}

return stack.back();

}

Ф-я перевода из постфиксной записи в инфиксную

string fromPostfixToPrefix(string postfix)

{

vector <string> stack;

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

if (postfix[i] == ' ') continue;

if (isOperator(postfix[i])) {

string operator1 = stack.back();

stack.pop\_back();

string operator2 = stack.back();

stack.pop\_back();

string temp = postfix[i] + operator2 + operator1;

stack.push\_back(temp);

}

else {

stack.push\_back(string(1, postfix[i]));

}

}

string result = "";

while (!(stack.empty())) {

result += stack.back();

stack.pop\_back();

}

return result;

}

Ф-я перевода из постфиксной записи в префиксную

float calculatePostfixExpression(string postfix)

{

vector <float> stack;

char symbol = 0;

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

symbol = postfix[i];

if (symbol == ' ') continue;

if (isOperator(symbol)) {

int x1 = stack.back();

stack.pop\_back();

int x2 = stack.back();

stack.pop\_back();

if (symbol == '+') stack.push\_back(x2 + x1);

else if (symbol == '-') stack.push\_back(x2 - x1);

else if (symbol == '\*') stack.push\_back(x1 \* x2);

else if (symbol == '/') stack.push\_back(x2 / x1);

}

else {

stack.push\_back(symbol - '0');

}

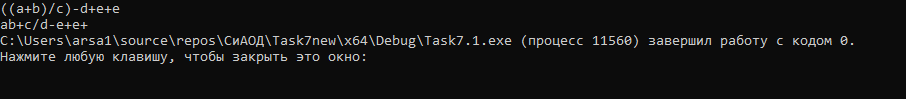
}

return stack.back();

}

Ф-я вычисления постфиксного выражения

## Вывод программы



## Общий вывод

Изучены способы записи арифмитических выражений (инфиксная, постфиксная, префиксная), разработы алгоритмы перевода между записями в различных формах

## Список информационных источников

1. Постфиксная, инфиксная, префиксная нотации: интернет
2. Переход из записей: лекции, интернет

# Практическая работа 8

## Цель работы:

* Получить навыки по анализу вычислительной сложности нескольких алгоритмов сортировки и определение наиболее эффективного алгоритма

## Задание

1. Постановка задачи: Разработать три алгоритма сортировки, определенные вариантом. Провести анализ вычислительной и емкостной сложности алгоритма на массивах, заполненных случайно. Определить наиболее эффективный алгоритм.
2. Модель решения: создание функций сортировок согласно алгоритму, создание массива на n элементов, тест сортировок на разном количестве элементов

## Реализация

void SelectionSort(int \*arr, int size)

{

unsigned int start\_time = clock();

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

int min\_index = i;

for (int j = i + 1; j < size; j++) {

if (arr[j] < arr[min\_index]) {

min\_index = j;

}

}

int temp = arr[i];

arr[i] = arr[min\_index];

arr[min\_index] = temp;

}

unsigned int end\_time = clock();

cout << (long double)(end\_time - start\_time) / CLOCKS\_PER\_SEC << endl;

}

Ф-я сортировки выбором. Добавлены строчки кода для определения времени исполнения

void ShakerSort(int\* arr, int start, int size)

{

unsigned int start\_time = clock();

int Left, Right, i;

Left = start;

Right = size - 1;

while (Left <= Right)

{

for (i = Right; i >= Left; i--) {

if (arr[i - 1] > arr[i]) swap(arr, i);

}

Left++;

for (i = Left; i <= Right; i++) {

if (arr[i - 1] > arr[i]) swap(arr, i);

}

Right--;

}

unsigned int end\_time = clock();

cout << (long double)(end\_time - start\_time) / CLOCKS\_PER\_SEC << endl;

}

Ф-я сортировки шейкерной.

void heapify(int\* numbers, int root, int bottom)

{

int maxChildIndex;

bool end = false;

while ((root \* 2 <= bottom) && (!end))

{

if (root \* 2 == bottom) maxChildIndex = root \* 2;

else if (numbers[root \* 2] > numbers[root \* 2 + 1]) maxChildIndex = root \* 2;

else maxChildIndex = root \* 2 + 1;

if (numbers[root] < numbers[maxChildIndex])

{

int temp = numbers[root];

numbers[root] = numbers[maxChildIndex];

numbers[maxChildIndex] = temp;

root = maxChildIndex;

}

else {

end = true;

}

}

}

void HeapSort(int\* arr, int size)

{

unsigned int start\_time = clock();

for (int i = (size / 2); i >= 0; i--)

heapify(arr, i, size - 1);

for (int i = size - 1; i >= 1; i--)

{

int temp = arr[0];

arr[0] = arr[i];

arr[i] = temp;

heapify(arr, 0, i - 1);

}

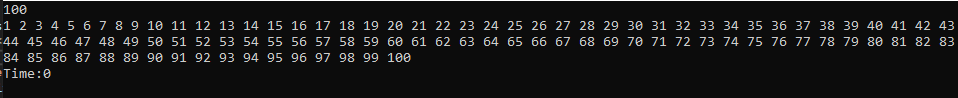
unsigned int end\_time = clock();

cout << (long double)(end\_time - start\_time) / CLOCKS\_PER\_SEC << endl;

}

Ф-я пирамидальной сортировки.

## Вывод программы



## Общий вывод

Были разработаны алгоритмы сортировок различной эффективности, оценена их скорость и эффективность для различного количества элементов в массиве.

## Список информационных источников

1. Алгоритмы сортировки: <habr.com>, интернет

# Практическая работа 9

## Цель работы:

* Получить практический опыт по применению алгоритмов поиска в таблицах данных.

## Задание

1. Постановка задачи: разработать и реализовать алгоритмы поиска записей с заданным ключом в таблице с применением различных алгоритмов.
2. Задание варианта:

|  |  |  |
| --- | --- | --- |
| 19 | Фибоначчи поиск | Регистрация земельного участка в СНТ: кадастровый номер – семизначное число, адрес СНТ |

1. Модель решения: Создание структуры LandArea с полями unsigned int для кадастрового номера и string для адреса Создание алгоритмов линейного поиска: перебором и с барьером. Разработка функций. Разработка функции Фибоначчи поиска

## Реализация

struct LandArea

{

unsigned int cadastral\_number = 0;

string address;

};

void bruteForceSearch(LandArea\* lands, int size, int query)

{

for (int i = 0; i < size; i++)

{

if (lands[i].cadastral\_number == query) {

cout << "Found: address - " << lands[i].address << endl;

}

}

}

void barrierSeach(LandArea\* lands, int size, int query)

{

LandArea last = lands[size - 1];

lands[size - 1].cadastral\_number = query;

int i = 0;

for (i; lands[i].cadastral\_number != query; ++i);

lands[size - 1] = last;

if (i != (size - 1) || query == last.cadastral\_number) {

cout << "Found: address - " << lands[i].address << endl;

}

}

bruteForceSearch производит поиск элемента в таблице перебором. barrierSearch – поиск с барьером

int fibonacchiSort(LandArea\* arr, int query, int size)

{

int fib1 = 0;

int fib2 = 1;

int fibM = fib1 + fib2;

while (fibM < size) {

fib1 = fib2;

fib2 = fibM;

fibM = fib2 + fib1;

}

int offset = -1;

while (fibM > 1) {

int i = min(offset + fib1, size - 1);

if (arr[i].cadastral\_number < query) {

fibM = fib2;

fib2 = fib1;

fib1 = fibM - fib2;

offset = i;

}

else if (arr[i].cadastral\_number > query) {

fibM = fib1;

fib2 = fib2 - fib1;

fib1 = fibM - fib2;

}

else

return i;

}

if (fib1 && arr[offset + 1].cadastral\_number == query)

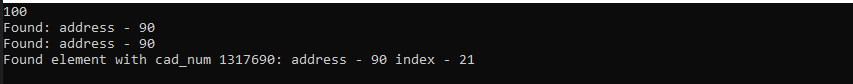
return offset + 1;

return -1;

}

Фибонначи поиск работает на отсортированном массиве, поэтому также была добавлена функция сортировки выбором из прошлых работ.

## Вывод программы



## Общий вывод

Изучены алгоритмы поиска элементов по ключу в таблице.

## Список информационных источников

1. Интернет

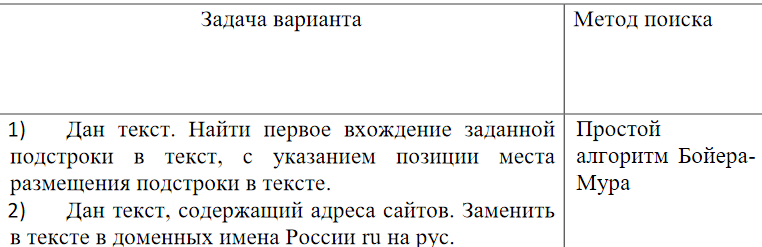
# Практическая работа 10

## Цель работы:

* Получить навыки и знания применения алгоритмов поиска в тексте (образца)

## Задание

1. Постановка задачи: Разработать и реализовать алгоритм поиска согласно варианту для задачи 1. Разработать и реализовать поиск образца в тексте с применением регулярных выражений для второй задачи варианта.
2. Модель решения: разработка алгоритма, создание функций для работы с текстом, изучение регулярных выражение, создание функции для обработки текста с использованием регулярных выражений.
3. Задание варианта:



## Реализация

// Функция поиска индекса первого вхождения строки temp в source, алгоритм поиска Бойера-Мура

// Возврат -1, если вхождения не нашлось, иначе индекс вхождения

int getFirstEntry(string source, string temp)

{

int sourceLen = source.length();

int tempLen = temp.length();

if (tempLen > sourceLen) return -1;

map<char, int> offsetTable;

for (int i = 0; i <= 255; i++) {

offsetTable[(char)i] = tempLen;

}

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

offsetTable[(char)temp[i]] = tempLen - i - 1;

}

int i = tempLen - 1, j = i, k = i;

while (j >= 0 && i <= sourceLen - 1) {

j = tempLen - 1;

k = i;

while (j >= 0 && (char)source[k] == (char)temp[j]) {

k--;

j--;

}

i += offsetTable[(char)source[i]];

}

if (k >= sourceLen - tempLen) {

return -1;

}

else {

return k + 1;

}

}

Функция поиска в тексте по алгоритму Бойера-Мура (упрощенный вариант).

// Ф-я изменения домена .ru на .рус

string changeDomen(string source)

{

setlocale(LC\_ALL, "Russian");

string to\_change = source;

vector <string> words\_to\_change;

regex pattern("[a-zа-я]{1,}.ru"); // шаблон для адресов

smatch result;

while (regex\_search(source, result, pattern)) {

words\_to\_change.push\_back(result.str());

source = result.suffix().str();

}

for (auto& word : words\_to\_change) {

int ind = to\_change.find(word);

if (ind != -1) to\_change.replace(to\_change.begin() + ind + word.length() - 3, to\_change.begin() + ind + word.length(), ".рус ");

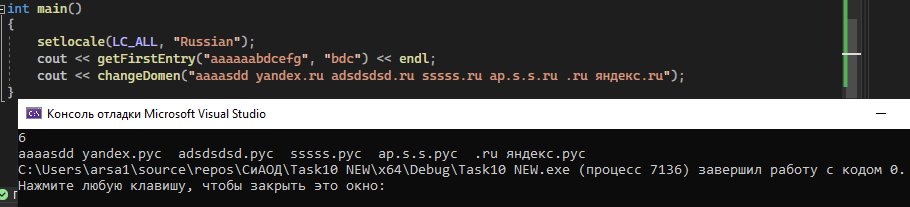
}

return to\_change;

}

Функция замены домена «.ru» на «.рус» в URL-адресе сайта с использованием регулярных выражений (библиотека regex).

## Вывод программы



## Общий вывод

На практике получены знания и навыки разработки эффективных алгоритмов поиска для поиска образца в тексте, получены навыки работы с регулярными выражениями.

## Список информационных источников

1. Интернет ([cppreference.com](http://www.cppreference.com) и другие)
2. Лекции