

МИНОБРНАУКИ РОССИИ

*Федеральное государственное бюджетное образовательное учреждение высшего образования*

***«МИРЭА – Российский технологический университет»***

**РТУ МИРЭА**

Отчет по выполнению практического задания № 9

**Тема**: **Алгоритмы поиска в массивах (таблицах)**

Дисциплина: Структуры и алгоритмы обработки данных

|  |  |
| --- | --- |
| Выполнил студент | Дмитриев П. В. |
|  | Фамилия И.О. |
| Группа | ИКБО-33-21 |
|  | Номер группы |

Москва 2022

Вариант № 7

Тема. Алгоритмы поиска в таблице (массиве).

Цель. Поучить практический опыт по применению алгоритмов поиска в таблицах данных.

Задание. Разработать и реализовать алгоритмы поиска записей с заданным ключом в таблице с применением различных алгоритмов. Т.е. выполнить два задания и оформить по ним отчеты в соответствии с требованиями, представленными в этом документе.

Задание 2. Разработать программу поиска записи по ключу в таблице записей с применением алгоритма, определенного в задании варианта.

1. Таблица содержит записи, структура которых определена вариантом. Ключи уникальны в пределах таблицы.

2. Разработать алгоритм поиска, определенный в варианте. Реализовать алгоритм функцией.

3. Провести практическую оценку времени выполнения алгоритмов на таблицах объемом 100, 1000, 10 000 записей на случайно заполненных таблицах (худший случай). На таблицах с лучшим временем и средним.

4. Составить таблицу с указанием: времени выполнения алгоритма, его фактическую и теоретическую вычислительную сложность.

ЗАДАНИЕ 1

1. **Разработка программы задачи 1**
   1. **Постановка задачи**

ВАРИАНТ 7

****

Разработать программу поиска записи по ключу в таблице записей с применение двух алгоритмов линейного поиска

1. Таблица содержит записи, структура которых определена вариантом. Ключи уникальны в пределах таблицы.

2. Разработать функцию линейного поиска (метод грубой силы).

3. Разработать функцию поиска с барьером.

4. Провести практическую оценку времени выполнения алгоритмов на таблицах объемом 100, 1000, 10 000 записей.

5. Составить таблицу с указанием: времени выполнения алгоритма, его фактическую и теоретическую вычислительную сложность.

6. Сделать выводы об эффективности алгоритмов.

**1.2 Реализация задания**

Создаем структуру данных spec с полями name – имя вуза, kod – код специальности

С помощью цикла for вводим данные о вузе и специальности. После заполнения данных, программа запрашивает на ввод число в переменную code – ключ, с помощью которого нужно найти название вуза. Далее запускается цикл for, внутри которого сравнивается поле kod элемента массива arr с введенным code, если они равны, то флаг j становится равным 1. Если j = 1, то выводится название вуза, если j = 0, то вуз не найден.

**Линейный поиск**

#include <iostream>

using namespace std;

struct spec {

string name;

int kod;

} arr[100];

int main() {

setlocale(LC\_ALL, "ru");

int code;

int j = 0;

for (int i = 0; i < 3; i++) {

cout << "Введите название вуза: ";

cin >> arr[i].name;

cout << "Введите код специальности: ";

cin >> arr[i].kod;

}

cout << "\n" << "Поиск вуза по коду специальности: ";

cin >> code;

for (int i = 0; i < 100; i++) {

if (arr[i].kod == code) {

j = 1;

if (j == 1) {

cout << "\nВуз с этой специальностью: " << arr[i].name << "\n";

}

}

}

if (j == 0) {

cout << "Вуз не найден" << "\n";

}

return 0;

}

**Поиск с барьером**

#include <iostream>

using namespace std;

struct spec {

string name;

int kod;

} arr[100];

int main() {

setlocale(LC\_ALL, "ru");

int code;

int j = 0;

for (int i = 0; i < 3; i++) {

cout << "Введите название вуза: ";

cin >> arr[i].name;

cout << "Введите код специальности: ";

cin >> arr[i].kod;

}

cout << "\n" << "Поиск вуза по коду специальности: ";

cin >> code;

for (int i = 0; i < 100; i++) {

if (arr[i].kod == code) {

cout << "\nВуз с этой специальностью: " << arr[i].name << "\n";

}

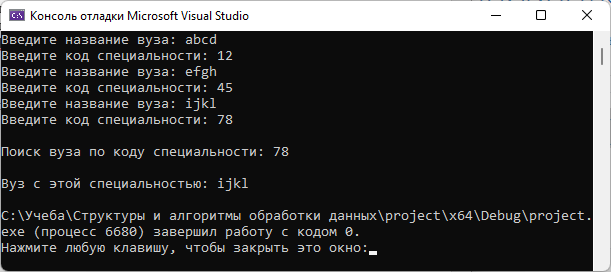
}

return 0;

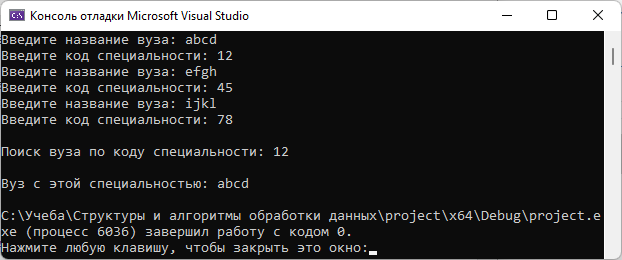
}

* 1. **Скриншоты**

**Линейный поиск**

****

**Поиск с барьером**

****

**Практическая оценка времени выполнения алгоритмов**

Для оценки разработаем другую программу, содержащую массив из цифр

#include <iostream>

#include <chrono>

using namespace std;

int main() {

setlocale(LC\_ALL, "ru");

srand(time(NULL));

int\* index;

int j = 0;

int\* arr, n;

int key;

cout << "Введите количество элементов массива: ";

cin >> n;

index = arr = new int[n];

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

arr[i] = 1 + rand() % n;

cout << arr[i] << " ";

}

cout << endl << "Введите число из массива: ";

cin >> key;

auto start = chrono::high\_resolution\_clock::now();

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

if (arr[i] == key) {

continue;

}

}

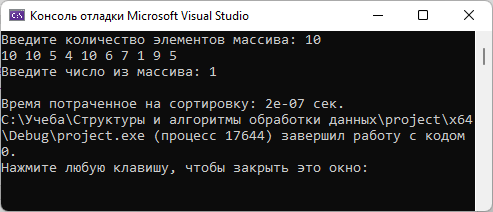
auto end = chrono::high\_resolution\_clock::now();

chrono::duration<float> duration = end - start;

cout << "\n" << "Время потраченное на сортировку: " << duration.count() << " сек.";

}

Скриншот



|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **T(n) в сек** | **T=f(C+M)** | **T=Cф+Мф** |
| 100 | 0.0000004 | O(100) | 100 |
| 1000 | 0.0000017 | O(1000) | 1000 |
| 10000 | 0.0000281 | O(10000) | 10000 |

Таблица 1 – Сводная таблица результатов

График зависимости времени выполнения алгоритма от объема обрабатываемых данных.

ЗАДАНИЕ 2

**2.1 Реализация задания**

Разработать программу поиска записи по ключу в таблице записей с применением алгоритма, определенного в задании варианта.

1. Таблица содержит записи, структура которых определена вариантом. Ключи уникальны в пределах таблицы.

2. Разработать алгоритм поиска, определенный в варианте. Реализовать алгоритм функцией.

3. Провести практическую оценку времени выполнения алгоритмов на таблицах объемом 100, 1000, 10 000 записей на случайно заполненных таблицах (худший случай). На таблицах с лучшим временем и средним.

4. Составить таблицу с указанием: времени выполнения алгоритма, его фактическую и теоретическую вычислительную сложность.

**2.2 Реализация задания**

Задание варианта: Интерполяционный поиск

Структура: Специализация вуза: код специальности, название вуза

В функции func() реализуется цикл while интерполирующего поиска. Поиск производит оценку области поиска по расстоянию между ключом поиска и текущим значением элемента. Если значение в ячейке с индексом mid меньше значения ключа, то смещаем нижнюю границу. В том случае, если значение в mid больше, то смещаем верхнюю границу. Иначе возвращаем индекс mid. Если цикл не вернул индекс искомого значения, то проверяем не находится ли оно в ячейке массива с индексом left, иначе возвращаем -1 (значение не найдено).

Переменной res присваивается значение возврата функции func(). На экран выводится name элемента res в массиве arr.

**2.3 Код программы**

#include <iostream>

using namespace std;

struct spec {

string name;

int kod;

} arr[6];

int func(spec\* massiv, int key) {

int mid, left = 0, right = 2;

//цикл интерполирующего поиска

while (massiv[left].kod <= key && massiv[right].kod >= key)

{

if (massiv[right].kod == massiv[left].kod)

break;

//интерполирующий поиск производит оценку новой области поиска

//по расстоянию между ключом поиска и текущим значение элемента

mid = left + ((key - massiv[left].kod) \* (right - left)) / (massiv[right].kod - massiv[left].kod);

//если значение в ячейке с индексом mid меньше, то смещаем нижнюю границу

if (massiv[mid].kod < key)

left = mid + 1;

//в случае, если значение больше, то смещаем верхнюю границу

else if (massiv[mid].kod > key)

right = mid - 1;

//если равны, то возвращаем индекс

else

return mid;

}

//если цикл while не вернул индекс искомого значения,

//то проверяем не находится ли оно в ячейке массива с индексом low,

//иначе возвращаем -1 (значение не найдено)

if (massiv[left].kod == key)

return left;

if (massiv[right].kod == key)

return right;

return -1;

}

int main() {

setlocale(LC\_ALL, "ru");

int code, res = 0;

int j = 0;

for (int i = 0; i < 3; i++) {

cout << "Введите название вуза: ";

cin >> arr[i].name;

cout << "Введите код специальности: ";

cin >> arr[i].kod;

}

cout << "\n" << "Поиск вуза по коду специальности: ";

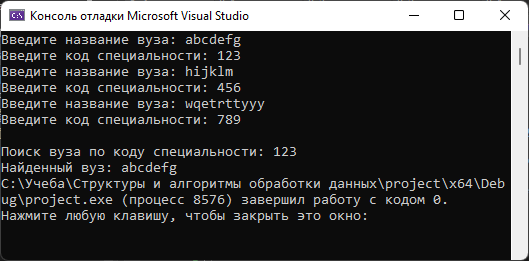
cin >> code;

res = func(arr, code);

cout << "Найденный вуз: " << arr[res].name;

}

**2.4 Скриншоты**

****

Время выполнения алгоритма интерполяционного поиска при n = 3: 0.0000004 сек.