**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЯ ОБРАЗОВАНИЯ**

**“ПОЛОЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ ЕВФРОСИНИИ ПОЛОЦКОЙ”**

Факультет информационных технологий

Кафедра технологий программирования

**Отчёт по Лабораторной работе №2**

**«Организация таблиц идентификаторов»**

Выполнил студент 2 курса, группа 21-ИТ-1 Макеёнок Д.И.

Проверил Сыцевич Д.Н.

Полоцк, 2022 г.

**Цель работы:** изучить основные методы организации таблиц

идентификаторов, получить представление о преимуществах и недостатках,

присущих различным методам организации таблиц идентификаторов.

Для выполнения лабораторной работы требуется написать программу,

которая получает на входе набор идентификаторов, организует таблицы

идентификаторов с помощью заданных методов, позволяет осуществить

многократный поиск произвольного идентификатора в таблицах и сравнить

эффективность методов организации таблиц. Список идентификаторов

считать заданным в виде текстового файла. Длина идентификаторов

ограничена 32 символами.

**Результат работы:**

#include<iostream>

#include<string>

using namespace std;

struct Node {

string val;

Node\* next;

Node(string \_val) : val(\_val), next(nullptr) {}

};

struct list {

Node\* first;

Node\* last;

list() : first(nullptr), last(nullptr) {}

bool is\_empty() {

return first == nullptr;

}

void push\_back(string \_val) {

Node\* p = new Node(\_val);

if (is\_empty()) {

first = p;

last = p;

return;

}

last->next = p;

last = p;

}

void print() {

if (is\_empty()) return;

Node\* p = first;

while (p) {

cout << p->val << ", ";

p = p->next;

}

}

Node\* find(string \_val) {

Node\* p = first;

while (p && p->val != \_val) p = p->next;

return (p && p->val == \_val) ? p : nullptr;

}

Node\* operator[] (const int index) {

if (is\_empty()) return nullptr;

Node\* p = first;

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

p = p->next;

if (!p) return nullptr;

}

return p;

}

};

int hash\_function(string str);

int main()

{

int const AMOUNT = 100;

string hash\_table1[AMOUNT];

list l[AMOUNT];

string str;

int id = hash\_function(str),mem\_id = 0, n = 1;

int\* pointer;

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

{

for (int j = 0; j < 5; j++) str += (rand() % 26) + 97;

id = hash\_function(str);

mem\_id = id;

while (1) //решение колизий с использованием псевдослучайных чисел

{

if (hash\_table1[id] == "")

{

hash\_table1[id] = str;

break;

}

else id = rand() % 100;

}

while (1) //решение колизий с использованием методом цепочки

{

if (l[mem\_id].is\_empty())

{

l[mem\_id].push\_back(str);

break;

}

else {

l[mem\_id].push\_back(str); //если айди занят,то значения записываються в список

break;

}

}

str = "";

}

/\*for (int i = 0; i < AMOUNT; i++)

{

if (l[i].is\_empty()){

} //Вывод списков в цепочном методе

else { l[i].print(); cout << "| "; }

}

cout << endl;\*/

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

{

if (hash\_table1[i] != "") cout << hash\_table1[i] << ", ";

}

cout << endl;

string search = ""; cin >> search;

n = 0; id = hash\_function(search); mem\_id = id;

int counter = 0;

while (1)

{

if (search == hash\_table1[id])

{

cout << "random" << endl;

cout << "\t" << search << " was Found" << endl;

cout << "\tCounts = " << counter + 1 << endl;

break;

}

else id = rand() % 100;

counter++;

}

counter = 0;

while (1) {

if (l[mem\_id].is\_empty()) {

}

else {

cout << "chains" << endl;

cout << "\t"; l[mem\_id].print(); cout << "List with desired value" << endl;

cout << "\tCounts = " << counter + 1 << endl;

break;

}

counter++;

}

}

int hash\_function(string str)

{

int id = 0;

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

{

id += str[i];

}

id %= 100;

return id;

}

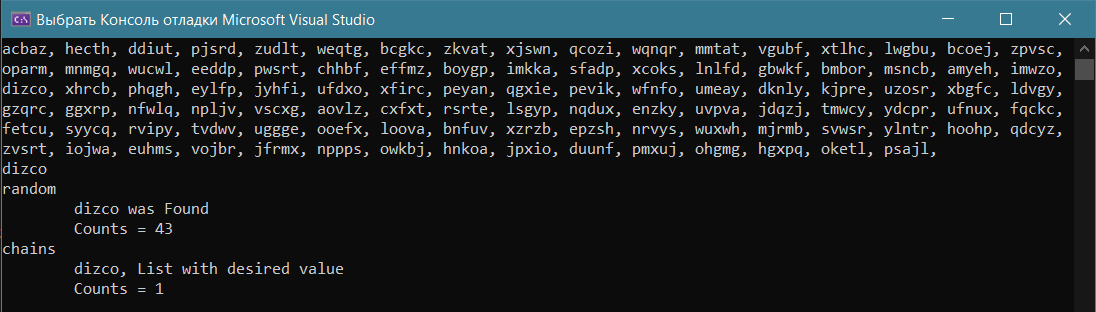


Рисунок 1 – Результат работы программы

**Вывод:** В результате выполнения написанного программного кода для ряда тестовых файлов было установлено, что при заполнении таблицы идентификаторов для поиска и размещения идентификатора с использованием рехэширования на основе генератора псевдослучайных чисел в среднем требуется больше число сравнений, чем при использовании метода цепочек.