**Лабораторная работа №5**

**Хеширование данных. Поиск данных в хеш-таблице.**

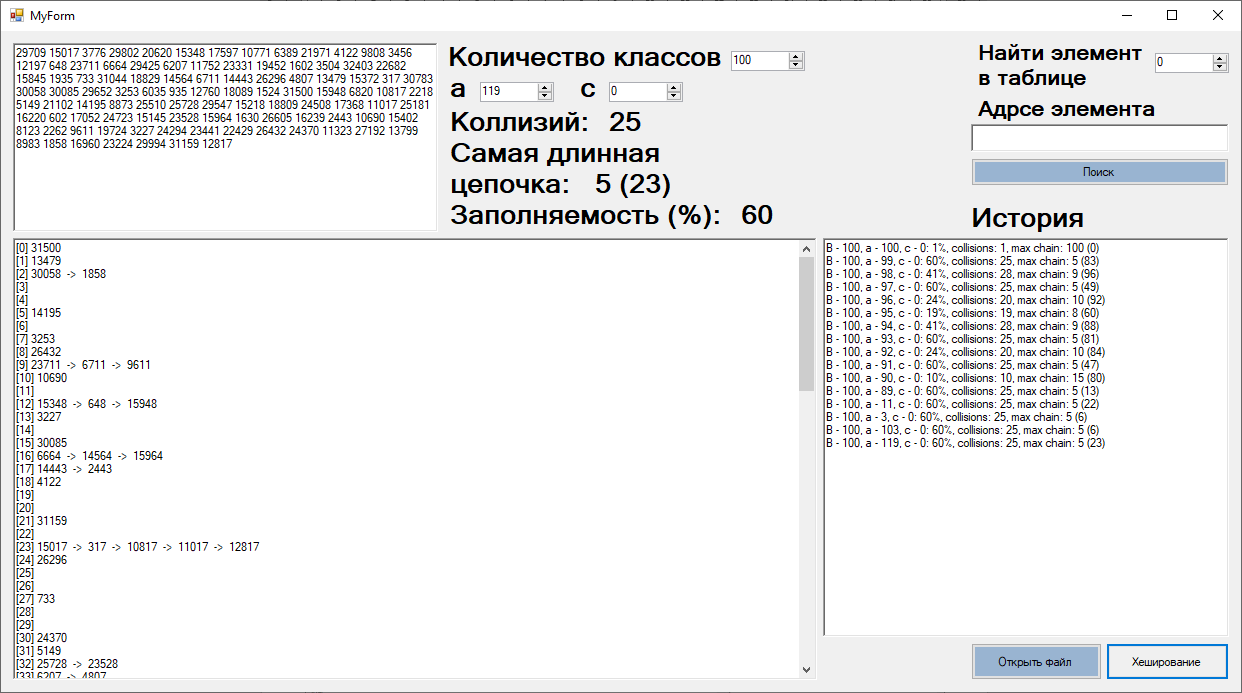
Разработать программу хеширования данных из файла методом открытого хеширования. Хеш-таблицу закодировать как класс, а операции с ней - как функции-члены класса. В программе предусмотреть следующий действия:

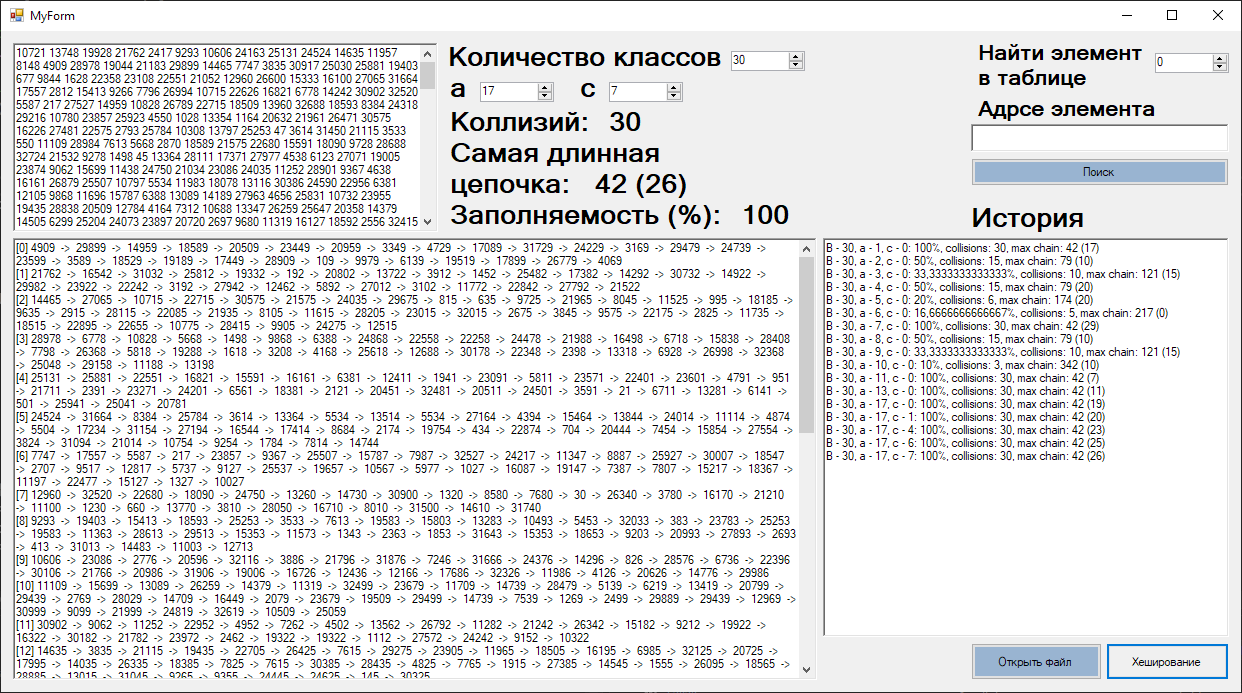
а) создание хеш-таблицы с небольшим количеством классов (B<=30) и небольшим набором исходных данных, вывод всей таблицы на экран, осуществление поиска по таблице, в качестве хеш-функции выбрать h(x)=x%B;

б) создание хеш-таблицы, для которой количество классов B задается пользователем (до 20000 - 50000), осуществление поиска по таблице, подсчет общего числа коллизий в таблице, нахождение самой длинной цепочки коллизий, процента заполняемости таблицы. В качестве хеш-функции выбрать h(x)=(ax+c)%B, проанализировать результаты заполняемости таблицы для различных a и c.

В качестве исходных данных выбрать: Файл целых чисел (варианты 1,4,7,10).

* Создать класс Hash\_table:
  + В public:
    - конструктор с инициализацией колмчества классов, параметров a и c; обнуление количества коллизий, максимальной цепочки, заполняемости
    - деструктор (удаление всей таблицы)
    - функция добавления элемента
    - функция записи всей хеш-таблицы в файл
    - геттеры количества коллизий, максимальной цепочки, индекса максимальной цепочки, заполняемости
  + В private: переменные h (значение хеш-функции), a, c, max\_chain (максимальная длина цепочки), index\_max\_chain (номер класса, где самая длинная цепочка), collisions (число коллизий), occupancy (число непустых классов), переменная для хранения количества классов, класс Queue хранения линейных списков (цепочек) – очередь и массив указателей класса Queue. В классе Queue предусмотреть:
    - В public:
      * конструктор (head = NULL, tail = NULL, chain = 0), деструктор, функции проверки на пустоту, добавления элемента в список, вывода списка, обнуления, геттер длины цепочки
    - В private: структуру Node для хранения целого числа и указателя на следующую ноду, «тело», «голову», переменную длины цепочки;
* Создать возможность открывать текстовые файлы и копировать их содержимое в программу;
* Выделить из файла нужные данные (целые числа) с помощью функции Split();
* Предусмотреть возможность задания значений a и c и количества классов (до 50000) пользователем;
* Загрузить данные в объект класса Hash\_table;
* Визуализировать хеш-таблицу (по желанию);
* Вывести на экран: число коллизий, длину самой длинной цепочки, процент заполняемости таблицы;
* Создать функцию для поиска в таблице элемента с указанием его адреса: номер класса, индекс в списке;
* Создать «Историю» хеширований: количество классов, a, c, процент заполняемости





Анализ:

1. «c» не влияет на заполняемость
2. «a» существенно влияет на заполняемость – если a и B взаимно простые числа (НОД=1), то заполняемость 100%, иначе заполняемость = 100% / НОД(a, B)

#include "MyForm.h"

#include <iostream>

using namespace System;

using namespace System::Windows::Forms;

using namespace System::IO;

using namespace System::ComponentModel;

using namespace System::Drawing;

[STAThread]

int main(array<String^>^ args)

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Lab5::MyForm form;

Application::Run(% form);

}

class Hash\_table {

public:

Hash\_table(int numericUpDown, int a, int c)

: numericUpDown(numericUpDown), a(a), c(c) {

collisions = 0; max\_chain = 0; occupancy = 0;

}

~Hash\_table() {

for (size\_t i = 0; i < numericUpDown; i++) {

B[i].nullQueue();

}

delete[] B;

}

void add\_element(int x) {

h = (a \* x + c) % numericUpDown;

B[h].add(x);

}

void print\_table() {

StreamWriter^ file = gcnew StreamWriter("hash.txt", false);

for (size\_t i = 0; i < numericUpDown; i++) {

file->Write(B[i].print());

if (B[i].get\_chain() > 0) occupancy++;

if (B[i].get\_chain() > 1) collisions++;

if (B[i].get\_chain() > max\_chain) {

max\_chain = B[i].get\_chain();

index\_max\_chain = i;

}

}

file->Close();

}

int get\_collisions()const { return collisions; }

int get\_max\_chain()const { return max\_chain; }

int get\_index\_max\_chain()const { return index\_max\_chain; }

double get\_occupancy()const { return occupancy; }

private:

long int h;

int a; int c;

int numericUpDown;

double occupancy;

int max\_chain;

int index\_max\_chain;

int collisions;

class Queue {

public:

Queue() {

head = NULL;

tail = NULL;

chain = 0;

}

~Queue() {}

//геттер длины цепочки

int get\_chain()const { return chain; }

//проверка на пустоту

bool empty() { return head == NULL; }

//добавление элемента

void add(int value) {

if (empty()) {

head = new Node;

head->data = value;

head->next = NULL;

tail = head;

chain = 1;

}

else {

tail->next = new Node;//при a от 3 и выше ошибка записии или чтения, с маленькими текстовыми файламии проблем нет

tail = tail->next;

tail->data = value;

tail->next = NULL;

chain++;

}

}

//вывод очереди

String^ print() {

String^ line = "";

if (!empty()) {

Node\* tmp = head;

while (head != NULL) {

line += head->data + " ";

head = head->next;

}

head = tmp;

}

return line + "\n";

}

//удаление очереди

void nullQueue() {

Node\* tmp;

while (!empty()) {

tmp = head;

head = head->next;

delete(tmp);

}

}

private:

struct Node {

int data;

Node\* next;

};

Node\* head, \* tail;

//жлина цепочки

int chain;

};

Queue\* B = new Queue[numericUpDown];

};

//загрузка текстового файла

System::Void Lab5::MyForm::file\_open\_Click(System::Object^ sender, System::EventArgs^ e)

{

OpenFileDialog^ openFile1 = gcnew OpenFileDialog;

openFile1->DefaultExt = "\*.txt";

openFile1->Filter = "Текстовые файлы (\*.txt)|\*.txt|Все файлы (\*.\*)|\*.\*";

if (openFile1->ShowDialog() == System::Windows::Forms::DialogResult::OK && openFile1->FileName->Length > 0) {

richTextBox1->Text = "";

richTextBox3->Text = "";

richTextBox\_log->Text = "";

richTextBox1->Text = System::IO::File::ReadAllText(openFile1->FileName, System::Text::Encoding::Default);

}

}

System::Void Lab5::MyForm::run\_Click(System::Object^ sender, System::EventArgs^ e)

{

if (numericUpDown1->Value != 0 && richTextBox1->Text != "") {

richTextBox3->Text = "";

array <String^>^ int\_file = richTextBox1->Text->Split(',', ' ', '\n', '\r', '\t', '\t');

Hash\_table hash(System::Convert::ToInt32(numericUpDown1->Value), System::Convert::ToInt32(numeric\_a->Value), System::Convert::ToInt32(numeric\_c->Value));

for (int i = 0; i < int\_file->Length; i++)

if (int\_file[i] != "" && int\_file[i] != " ")

hash.add\_element(System::Convert::ToInt32(int\_file[i]));

//запись хеш-таблицы в текстовый файл "hash.txt"

hash.print\_table();

//чтение хеш-таблицы из текстового файла в файл "hash\_table.txt"

StreamReader^ file = gcnew StreamReader("hash.txt");

StreamWriter^ file\_new = gcnew StreamWriter("hash\_table.txt", false);

for (size\_t i = 0; i < numericUpDown1->Value; i++) {

array <String^>^ hash = file->ReadLine()->Split(' ', '\n', '\r');

file\_new->Write("[" + i + "] ");

for (int j = 0; j < hash->Length; j++)

{

if (hash[j] != "" && hash[j] != " ") {

if (j != 0)file\_new->Write(" -> ");

file\_new->Write(hash[j]);

}

}

file\_new->Write("\n");

}

file->Close();

file\_new->Close();

StreamReader^ file\_table = gcnew StreamReader("hash\_table.txt");

richTextBox3->Text = file\_table->ReadToEnd();

file\_table->Close();

label\_percent->Text = System::Convert::ToString(hash.get\_occupancy() / System::Convert::ToInt32(numericUpDown1->Value) \* 100);

label\_collisions->Text = System::Convert::ToString(hash.get\_collisions());

label\_chain->Text = System::Convert::ToString(hash.get\_index\_max\_chain());

richTextBox\_log->Text += "B - " + numericUpDown1->Value + ", a - " + numeric\_a->Value + ", c - " + numeric\_c->Value + ": " + label\_percent->Text + "%" + Environment::NewLine;

MessageBox::Show(L"Таблица записана в файл 'hash\_table.txt'", L"Программа успешно выполнена", MessageBoxButtons::OK);

}

else {

MessageBox::Show(L"Неверное количество классов или пустая входная строка", L"Ошибка!", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

}

System::Void Lab5::MyForm::button\_search\_Click(System::Object^ sender, System::EventArgs^ e)

{

richTextBox4->Text = "";

int h = (System::Convert::ToInt32(numeric\_a->Value) \* System::Convert::ToInt32(numeric\_element->Value) + System::Convert::ToInt32(numeric\_c->Value)) % System::Convert::ToInt32(numericUpDown1->Value);

StreamReader^ file = gcnew StreamReader("hash.txt");

for (size\_t i = 0; i < h; i++)

file->ReadLine();

array <String^>^ number = file->ReadLine()->Split(' ', '\n', '\r');

for (size\_t i = 0; i < number->Length; i++)

{

if (number[i]!="" && System::Convert::ToInt32(numeric\_element->Value) == System::Convert::ToInt32(number[i])) {

richTextBox4->Text = h + " класс, " + i + " элемент";

file->Close();

return;

}

else if (i = number->Length - 1) {

MessageBox::Show(L"Элемент не найден", L"Программа успешно выполнена", MessageBoxButtons::OK);

}

}

file->Close();

}