МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Алгоритмы и структуры данных»

Тема: Хеш-таблицы с открытой адресацией

Студент гр. 8381 _____ Облизанов А.Д.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2019

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Облизанов А.Д.

Группа 8381

Тема работы: Хеш-таблицы с открытой адресацией

Исходные данные: необходимо провести исследование алгоритма вставки в хеш-таблицу с открытой адресацией, включающее генерацию входных данных, использование их для измерения количественных характеристик алгоритмов, сравнение экспериментальных результатов с теоретическими.

Содержание пояснительной записки:
«Содержание», «Введение», «Задание», «Описание программы», «Тестирование», «Исследование», «Заключение», «Список использованных источников».

Предполагаемый объем пояснительной записки:
Не менее 60 страниц.

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

Студент _____ Облизанов А.Д.

Преподаватель _____ Жангиров Т.Р.

АННОТАЦИЯ

В ходе выполнения курсовой работы была разработана программа с GUI, позволяющая исследовать алгоритм вставки в хеш-таблицу с открытой адресацией с различными хеш-функциями. Программа обладает следующей функциональностью: создание файла, содержащего входные данные для хеш-таблицы, создание хеш-таблицы по заданному файлу, хеш-функции и размеру хеш-таблицы; построение графиков зависимости числа итераций от нескольких параметров хеш-таблицы или входных данных для каждой хеш-функции. В результате исследования было выявлено, что выбор хеш-функции, длины таблицы, а также длина ключей элементов хеш-таблицы значительно влияют на скорость работы алгоритма в среднем случае.

SUMMARY

In the course work a program with a GUI was developed that allows you to explore with different hash functions the algorithm of inserting into a hash table with open addressing. The program has the following functionality: creating a file containing input data for a hash table, creating a hash table for a given file, hash function and hash table size; plotting the dependence of the number of iterations on several parameters of the hash table or input data for each hash function. The study showed that the choice of the hash function, the length of the table, as well as the length of the keys of the hash table elements significantly affect the speed of the algorithm in the average case.

СОДЕРЖАНИЕ

	Введение	6
1.	Задание	7
2.	Описание программы	8
2.1.	Описание интерфейса пользователя	8
2.2.	Описание основных классов для хеш-таблицы	10
2.3.	Описание алгоритма вставки в хеш-таблицу	12
2.4.	Описание класса StrStrWorker для работы с хеш-таблицей	12
2.5.	Описание структур и функций для исследования	14
3	Тестирование	16
3.1.	Вид программы	16
3.2.	Тестирование генерации файла	17
3.3.	Тестирование создания хеш-таблицы	18
4	Исследование	21
4.1	План экспериментального исследования	21
4.2	Технология проведения исследования	22
4.3	Исследование зависимостей от максимальной длины ключа	24
4.4	Исследование зависимостей от множителя размера таблицы	26
4.5	Исследование зависимостей от числа пар	29
4.6	Вывод об эффективности хеш-функций	34
4.7	Исследование худшего случая вставки	35
4.8	Выводы об исследовании алгоритма	37
	Заключение	38
	Список использованных источников	39
	Приложение А. Исходный код программы. main.c	40
	Приложение Б. Исходный код программы. pair.h	41
	Приложение В. Исходный код программы. cvector.h	46
	Приложение Г. Исходный код программы. hashtable.h	48
	Приложение Д. Исходный код программы. strstrworker.h	53

Приложение Е. Исходный код программы. strstrworker.cpp 54
Приложение Ж. Исходный код программы. strstrtester.h 60
Приложение И. Исходный код программы. strstrtester.cpp 61
Приложение К. Исходный код программы. mainwindow.h 65
Приложение Л. Исходный код программы. mainwindow.cpp 66
Приложение М. Исходный код программы. libraries.h 70
Приложение Н. Исходный код программы. lr5.pro 71
Приложение О. Исходный код программы. mainwindow.ui 72

ВВЕДЕНИЕ

Цель работы

Реализация и экспериментальное машинное исследование алгоритмов кодирования (Фано-Шеннона, Хаффмана), быстрого поиска на основе БДП или хеш-таблиц, сортировок.

Основные задачи

Генерация входных данных, использование их для измерения количественных характеристик структур данных, алгоритмов, действий, сравнение экспериментальных результатов с теоретическими.

Методы решения

Разработка программы велась на базе операционной системы Windows 10 в среде разработки QtCreator…

1. ЗАДАНИЕ

Необходимо провести исследование алгоритма вставки в хеш-таблицу с открытой адресацией в среднем и худшем случаях.

Исследование должно содержать:

1. Анализ задачи, цели, технологию проведения и план экспериментального исследования.
2. Генерацию представительного множества реализаций входных данных (с заданными особенностями распределения (для среднего и для худшего случаев)).
3. Выполнение исследуемых алгоритмов на сгенерированных наборах данных. При этом в ходе вычислительного процесса фиксируется как характеристики (например, время) работы программы, так и количество произведенных базовых операций алгоритма.
4. Фиксацию результатов испытаний алгоритма, накопление статистики.
5. Представление результатов испытаний, их интерпретацию и сопоставление с теоретическими оценками.

2. ОПИСАНИЕ ПРОГРАММЫ

2.1. Описание интерфейса пользователя

Интерфейс программы разделен на четыре части: панель ввода данных для генерации файла, панель ввода данных для создания хеш-таблицы, панель ввода данных для исследования алгоритма вставки в хеш-таблицу и построения графиков и панель вывода. Основные виджеты панели генерации файла и их назначение представлены в табл. 1.

Таблица 1 – Основные виджеты панели генерации файла

Класс объекта	Название виджета	Назначение
QTextEdit	GInputPairs	Поле ввода числа пар в файле
QTextEdit	GInputLength	Поле ввода максимальной длины ключа и значения
QTextEdit	GInputFile	Поле ввода названия файла
QCheckBox	GCheckWorst	Флаг выбора генерации худшего случая
QPushButton	GPushOk	Кнопка запуска генерации файла

Основные виджеты панели создания хеш-таблицы и их назначение представлены в табл. 2.

Таблица 2 – Основные виджеты панели создания хеш-таблицы

Класс объекта	Название виджета	Назначение
QTextEdit	HInputFile	Поле ввода файла для считывания
QTextEdit	HInputOut	Поле ввода файла для записи
QSpinBox	spinHFunc	Счетчик для выбора хеш-функции
QSpinBox	spinFactor	Счетчик для выбора множителя размера хеш-таблицы
QPushButton	HPushOk	Кнопка создания хеш-таблицы

Основные виджеты панели исследования и их назначение представлены в табл. 3.

Таблица 3 – Основные виджеты панели исследования

Класс объекта	Название виджета	Назначение
QRadioButton	MTestPairs	Флаг выбора исследования зависимости от числа пар
QRadioButton	MTestKey	Флаг выбора исследования зависимости от длины ключа
QRadioButton	MTestFactor	Флаг выбора исследования зависимости от размера таблицы
QRadioButton	MTestSum	Флаг выбора исследования суммы итераций вставок
QRadioButton	MTestMax	Флаг выбора исследования максимального числа итераций вставки
QTextEdit	MInputMin	Поле ввода минимального значения выбранного параметра
QTextEdit	MInputMax	Поле ввода максимального значения выбранного параметра
QSpinBox	spinStep	Счетчик для выбора шага изменения выбранного параметра
QPushButton	MPushOk	Кнопка запуска исследования

Для вывода графика используется сторонняя библиотека QCustomPlot. Основные виджеты панели вывода и их назначение представлены в табл. 4.

Таблица 4 - Основные виджеты панели вывода

Класс объекта	Название виджета	Назначение
QRadioButton	OutSetPlot	Флаг показа поля вывода графика
QRadioButton	OutSetText	Флаг показа текстового вывода
QCustomPlot	outputPlot	Поле вывода графика

Окончание таблицы 4

QTextEdit	outputText	Поле текстового вывода

2.2. Описание основных классов для хеш-таблицы

Для реализации хеш-таблицы были созданы шаблонные классы для пары элементов – Pair, и динамического массива – CVector. Класс Pair содержит два поля для элементов любого типа, а также флаги, используемые в хеш-таблице. Основные методы класса Pair представлены в табл. 5.

Таблица 5 – Основные методы класса Pair

Метод	Назначение
S getFirst();	Возвращает первый элемент пары типа S
T getSecond();	Возвращает второй элемент пары типа T
int isActive();	Возвращает true, если в пару заносились и не удалялись какие-либо значения
void setFirst(S);	Устанавливает первый элемент типа S
void setSecond(T);	Устанавливает второй элемент типа T
void setDeleted();	Устанавливает флаг, помечающий пару как ранее удаленную
const Pair & operator= (const Pair &other);	Копирует элементы пары (присваивание)

Динамический массив CVector для хэш-таблицы используется для хранения пар «ключ-значение», однако реализован также с помощью шаблонов. Основные методы класса приведены в табл. 6.

Таблица 6 – Основные методы класса CVector

Метод	Назначение
unsigned int getCapacity();	Возвращает текущий объем выделенной памяти для динамического массива в элементах

Окончание таблицы 6

int resize (unsigned int nsize);	Изменяет объем выделенной памяти, точнее выделяет память с новым объемом, копируя туда элементы массива
T & operator[] (unsigned int index);	Позволяет обращаться к элементу массива с помощью скобок [] по индексу
CVector<T> & operator= (const CVector<T> &);	Копирует элементы массива (присваивание)

Для реализации хеш-таблицы был создан шаблонный класс HashTable, который хранит в себе динамический массив пар и обеспечивает выполнение операции вставки. Основные метода класса приведены в табл. 7.

Таблица 7 – Основные методы класса HashTable

Метод	Назначение
void setSize(int size);	Изменяет объем выделенной памяти для таблицы (динамического массива)
int getSize();	Возвращает объем выделенной памяти для таблицы (количество пар)
Pair<S,T> & operator[] (unsigned int index);	Позволяет обращаться к элементу таблицы с помощью скобок [] по индексу
T & operator[] (unsigned int index);	Позволяет обращаться к элементу массива с помощью скобок [] по индексу
unsigned int set(S x, T y, unsigned int index);	Осуществляет вставку в таблицу по правилам открытой адресации
Pair<S,T> get(S key, unsigned int index);	Возвращает элемент таблицы (пару) по заданному ключу и индексу (хеш-функции)
unsigned int remove(S key, unsigned int index);	Удаляет элемент таблицы (пару) по заданному ключу и индексу (хеш-функции), помечая этот элемент как удаленный для реализации открытой адресации

Окончание таблицы 7

string print();	Возвращает строку, в которой содержится представление хеш-таблицы в формате: <индекс> (<ключ> <значение>)
string getString(S key, unsigned int index);	Возвращает строку, в которой содержится информация о паре в хеш-таблице по заданному ключу и индексу (хеш-функции)

2.3. Описание алгоритма вставки в хеш-таблицу

Хеш-таблица содержит массив, элементы которого есть пары. Выполнение вставки в хеш-таблицу начинается с вычисления хеш-функции. Алгоритм вставки элемента последовательно проверяет ячейки массива, начиная с ячейки с индексом значения хеш-функции, пока не будет найдена первая свободная ячейка, в которую и будет записан новый элемент. Используется линейное пробирование, то есть ячейки хеш-таблицы последовательно просматриваются друг за другом, с единичным интервалом. Пример коллизии с разрешением открытой адресацией представлен на рис. 1.

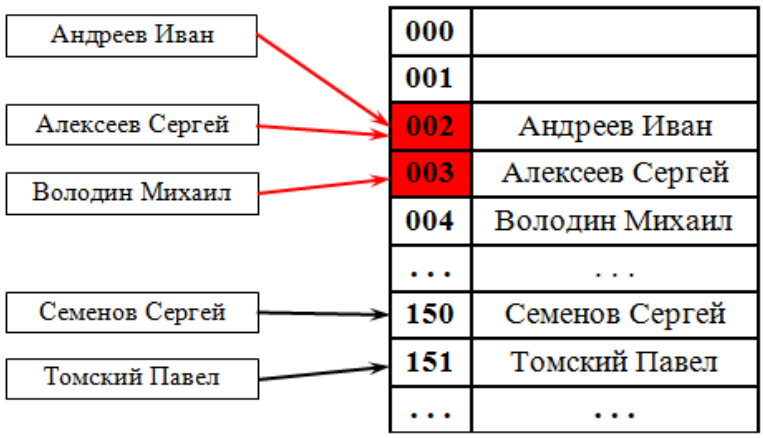

Рисунок 1 – Пример коллизии в открытой адресации

Здесь стрелки показывают значение хеш-функции для ключа. Так как у ключей «Андреев Иван» и «Алексеев Сергей» одинаковое значение хеш-функции, последний добавленный размещается на следующую ячейку. Так как ячейка массива по значению хеш-функции ключа «Володин Михаил» занята, то этот ключ также размещается на следующей ячейке.

2.4. Описание класса StrStrWorker для работы с хеш-таблицей

Для демонстрации работы алгоритмов хеширования, вставки в хеш-таблицу для ключей и значений был выбран формат строк. Для функций, отвечающих за генерацию файла входных данных, создание хеш-таблицы и формирование выходных данных, был создан класс StrStrWorker. В классе есть три метода, представляющие собой хеш-функции, которые генерируют индекс по строке-ключу. Алгоритм вычисления для каждой хеш-функции представлен в табл. 8.

Таблица 8 – Хеш-функции

Название функции	Алгоритм хеширования
hFunc1	Вычисляется как сумма кодов всех символов в ключе, берется остаток от деления на длину таблицы
hFunc2	Вычисляется как произведение суммы кодов первого и последнего символов ключа и его длины, берется остаток от деления на длину таблицы
hFunc3	Вычисляется как произведение суммы сложений кодов всех символов ключа и его длины, берется остаток от деления на длину таблицы

Класс имеет поле хеш-таблицы, в которой ключи и значения представлены строками, поле структуры hCreateInfo, которая хранит информацию о размере хеш-таблицы, количестве вставленных элементов, максимальном количестве итераций одной вставки, сумме количества итераций для всех вставок, а также флаг наличия ошибок. Основные методы класса StrStrWorker, за исключением хеш-функций, приведены в табл. 9.

Таблица 9 – Основные методы класса StrStrWorker

Метод	Назначение
QString getFromFile (QString fileName);	Возвращает содержимое файла по указанному пути в виде строки

Окончание таблицы 9

int generateFile (int size, int maxLength, QString fileName, bool worstCase, int hFuncNum);	Генерирует файл с различными случайными ключами с возможностью генерации худшего случая для заданной хеш-функции
hCreateInfo createHashTable(QString input, int hFuncNum, int factor, QString fileName)	Создает хеш-таблицу по входной строке, записывая информацию о таблице в заданный файл и возвращая основную информацию о процессе вставки элементов.
void saveStrToFile(QString output, QString fileName);	Записывает строку в указанный файл

2.5 Описание структур и функций для исследования

Исследование алгоритма вставки заключается в генерации файла и создания хеш-таблицы при разных значениях некоторых параметров. Информация, необходимая для начала исследования, содержится в структуре testInfo, поля которой представлены в табл. 10.

Таблица 10 – Поля структуры testInfo

Названия	Назначение
int mode	Может хранить следующие значения: 1 – если алгоритм исследуется при разных значениях длины ключа, 2 – при разных значениях количества пар, 3 – при разных значениях множителя длины таблицы
int min, max, step	Хранят соответственно минимальное, максимальное значение изменяемого параметра, а также шаг изменения
QString fileName	Название файла для вывода полученных данных
bool checkMaxIter	Отвечает за выбор величины для построения графика: максимальной итерации вставки, либо всех итераций
bool worstCase;	Отвечает за выбор худшего случая при генерации файла
int maxLength, size, factor	Хранят соответственно максимальную длину ключа, количество пар, множитель длины таблицы

Информация, получаемая в результате исследования, хранится в структуре testResult. Она включает в себя двумерный массив чисел сумм итераций вставок для трех хеш-функций, двумерный массив чисел максимальных итераций вставок для трех хеш-функций, а также массив значений выбранного параметра.

Исследование проводится функцией pairNumTester, которая принимает на вход объект класса StrStrWorker, структуру testInfo, ссылку на виджет QCustomPlot, а также ссылку на строку вывода.

Функция изменяет выбранный параметр от минимального до максимального значения с выбранным шагом. При каждом значении параметра генерируется файл, по нему создается хеш-таблица по каждой хеш-функции. Полученная информация о количестве максимального числа итераций вставки и общем количестве итераций алгоритма вставки сохраняется в структуру testResult, выводится в выходную строку в виде таблицы, а также добавляется в соответствующие графики точками. Описание функции приведено в приложении N.

3. ТЕСТИРОВАНИЕ

3.1. Вид программы

Программа представляет собой окно с графическим интерфейсом шириной 1280 пикселей и высотой 1000 пикселей. Вид программы после запуска представлен на рис. 2.

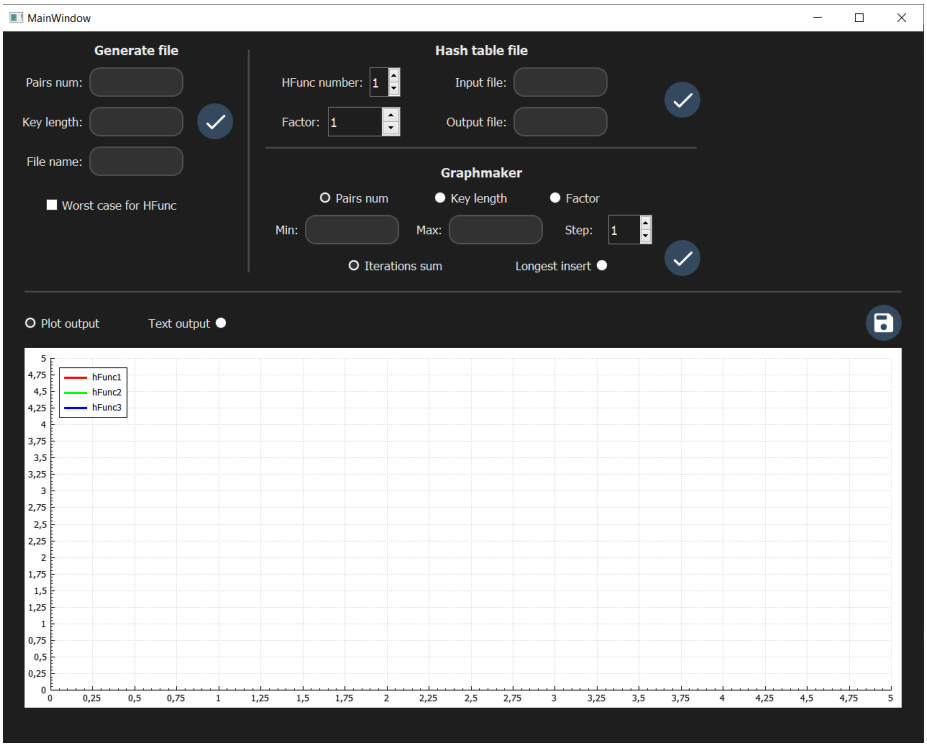

Рисунок 2 – Вид программы после запуска

После генерации файла выводятся сообщения об ошибке в входных данных, либо о успешном создании файла. После создания хеш-таблицы в файл вывода и в поле текстового вывода помещается информация о хеш-таблице: количество памяти, выделенной на таблицу (в элементах), число элементов, максимальное число итераций вставки, а также общее чисто итераций вставок, элементы хеш-таблицы. Вид программы после создания хеш-таблицы представлен на рис. 3.

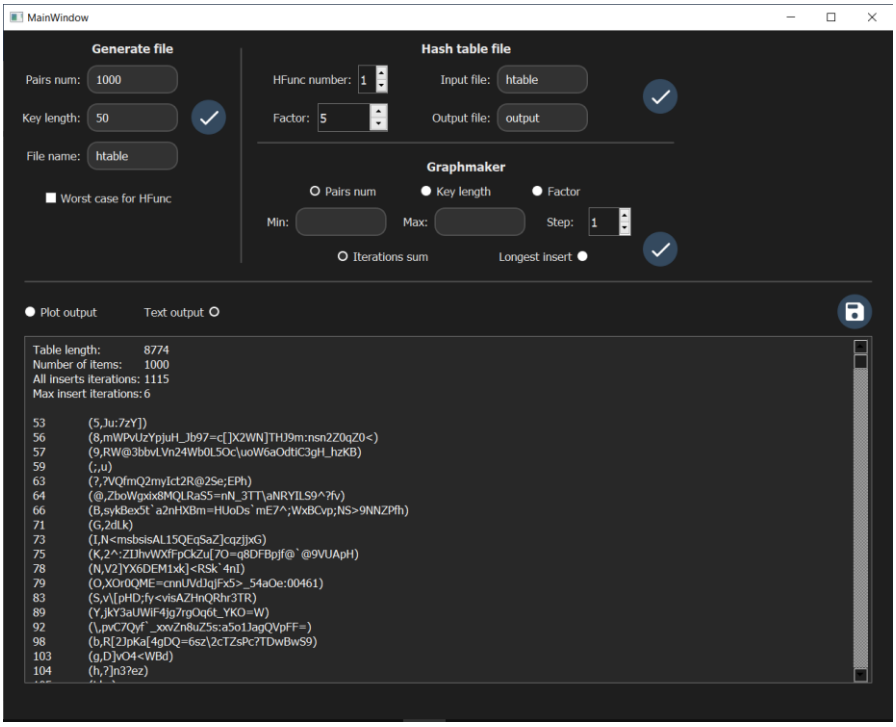

Рисунок 3 – Вид программы после создания хеш-таблицы

3.2. Тестирование генерации файла

Генерация файла была протестирована для различной длины ключа, числа пар, а также для худшего случая хеш-функций. Входные параметры: L – максимальная длина ключа, N – число пар. Результаты тестирования приведены в табл. 11.

Таблица 11 – Тестирование генерации файла

Худший случай	N	L	Содержание сгенерированного файла
Нет	3	5	9 RW@3 bvLV 24Wb0 5O \uoW
Нет	3	10	9 RW@3bbvL n24Wb0 5Oc\ oW6aOdtiC3 H_hzKByf

Окончание таблицы 11

Нет	3	2	9 RW 3 bv V 24
hFunc1	3	5	09hll W@3 bbrs LVn24 Wbxx L
hFunc1	3	10	09hRW@3bbPQ LVn24Wb0L5 Oc\uoW6ij OdtiC3g H_hzKByab h`5_rDPi
hFunc2	3	5	a0hWz 9R@ a3bLz bvV an4bz 2W0
hFunc3	3	15	09hRW@3bbvLVnbb 4 Wb0L5Oc\uoW6aXY dtiC3gH 4sUVGyUC6w5ULgg tWRv3

Из таблицы видно, что количество пар соответствует N, а длина ключа или значения не превышает M. В качестве символов используются английские буквы, некоторые символы и цифры. Правильность подбора худшего случая проще всего оценить для хеш-функции hFunc2, которая считается как произведение суммы кодов первого и последнего символов ключа на длину. Из таблицы видно, что ключи в худшем случае имеют одинаковые первые и последние символы, а их длина одинакова. Таким образом, все ключи будут давать одинаковое значение хеш-функции.

3.3. Тестирование создания хеш-таблицы

Хеш-таблица создается по сгенерированному файлу, имеется возможность выбора хеш-функции, а также множителя размера хеш-таблицы X. Изначальный размер хеш-таблицы H вычисляется по формуле $H = N * X$, где N – число пар в

файле. Для тестирования было выбрано значение $X = 5$. Результаты тестирования представлены в табл. 12.

Таблица 12 – Тестирование создания хеш-таблицы

Хеш-функ.	Содержимое файла	Данные о таблице
1	9 RW@3bbvLVn24 b0L5Oc\u W6aOdtiC3gH_h KByfh`5_rDPiSAD <Tst4s VGyUC6w5 LDtWRv3i n91cq_gfzrAF V\oNoD	Table length: 256 Number of items: 5 All inserts iterations: 5 Max insert iterations: 1 6 (VGyUC6w5,LDtWRv3i) 7 (9,RW@3bbvLVn24) 12 (b0L5Oc\u,W6aOdtiC3gH_h) 20 (KByfh`5_rDPiSAD,<Tst4s) 24 (n91cq_gfzrAF,V\oNoD)
1	09hRW@3bbvLVnbb 4 Wb0L5Oc\uoW6aXY dtiC3gH _hzKByfh`5_yy DPiSADJ<Tst4sU VGyUC6w5ULDtW]^ v3iin91	Table length: 256 Number of items: 4 All inserts iterations: 10 Max insert iterations: 4 15 (09hRW@3bbvLVnbb,4) 16 (Wb0L5Oc\uoW6aXY,dtiC3gH) 17 (_hzKByfh`5_yy,DPiSADJ<Tst4sU) 18 (VGyUC6w5ULDtW]^,v3iin91)
2	9 RW@3bbvLVn24 b0L5Oc\u W6aOdtiC3gH_h KByfh`5_rDPiSAD <Tst4s VGyUC6w5 LDtWRv3i n91cq_gfzrAF V\oNoD	Table length: 256 Number of items: 5 All inserts iterations: 6 Max insert iterations: 2 10 (n91cq_gfzrAF,V\oNoD) 12 (VGyUC6w5,LDtWRv3i) 14 (9,RW@3bbvLVn24) 20 (b0L5Oc\u,W6aOdtiC3gH_h) 21 (KByfh`5_rDPiSAD,<Tst4s)

Окончание таблицы 12

2	a0hW3bLn4z 9R@bvV2W abLO\o6Otz 05cuWadi aCg_zBf`_z 3HhKyh5r aDiAJTtsVz PSD<s4UG ayCwUDWviz U65LtR3i	Table length: 256 Number of items: 5 All inserts iterations: 15 Max insert iterations: 5 15 (a0hW3bLn4z,9R@bvV2W) 16 (abLO\o6Otz,05cuWadi) 17 (aCg_zBf`_z,3HhKyh5r) 18 (aDiAJTtsVz,PSD<s4UG) 19 (ayCwUDWviz,U65LtR3i)
3	9 RW@3bbvLVn24 b0L5Oc\u W6aOdtiC3gH_h KByfh`5_rDPiSAD <Tst4s VGyUC6w5 LDtWRv3i n91cq_gfzrAF V\oNoD	Table length: 256 Number of items: 5 All inserts iterations: 5 Max insert iterations: 1 0 (KByfh`5_rDPiSAD <Tst4s) 7 (9 RW@3bbvLVn24) 13 (n91cq_gfzrAF V\oNoD) 21 (b0L5Oc\u W6aOdtiC3gH_h) 23 (VGyUC6w5 LDtWRv3i)
3	09hRW@3bbvLVnbb 4 Wb0L5Oc\uoW6aXY dtiC3gH 4sUVGyUC6w5ULgg tWRv3 SE=;Zl2XUwhYrNN o;?e96D0 evA=Gr`;cLMU;aa o\wY;yN:Z	Table length: 256 Number of items: 5 All inserts iterations: 15 Max insert iterations: 5 0 (09hRW@3bbvLVnbb 4) 1 (Wb0L5Oc\uoW6aXY dtiC3gH) 2 (4sUVGyUC6w5ULgg tWRv3) 3 (SE=;Zl2XUwhYrNN o;?e96D0) 4 (evA=Gr`;cLMU;aa o\wY;yN:Z)

Из таблицы видно, что в худших случаях происходят множественные коллизии, элементы расположены последовательно друг за другом, число итераций в худшем случаи вставки совпадает с числом пар. В общем случае элементы располагаются хаотично и в случае отсутствия коллизий имеют индексы от 0 до $N * X$. Стоит отметить, что таблица имеет динамический размер и может осуществлять вставку пар по индексу, который больше исходного размера H.

4. ИССЛЕДОВАНИЕ

4.1. План экспериментального исследования.

Для проведения исследования сложности алгоритма вставки в хеш-таблицу необходимо выяснить параметры, от которых зависит эффективность алгоритма. Так как важной частью алгоритма является хеширование, то следует исследовать алгоритм с разными способами хеширования, определяя для каждого зависимость числа итераций и времени исполнения от выбранного параметра. После накопления данных необходимо провести сравнение результатов для различных хеш-функций и сделать выводы об эффективности алгоритмов хеширования. Кроме того, объединив всю статистику, следует сравнить полученные экспериментальные зависимости от теоретических и сделать выводы о сложности алгоритма вставки в среднем и худшем случаях.

План проведения исследования:

1. Сбор информации о зависимости числа итераций в среднем случае
   - 1.1. Зависимость от числа пар
     - 1.1.1. Для hFunc1
     - 1.1.2. Для hFunc2
     - 1.1.3. Для hFunc3
   - 1.2. Зависимость от максимальной длины ключа
     - 1.2.1. Для hFunc1
     - 1.2.2. Для hFunc2
     - 1.2.3. Для hFunc3
   - 1.3. Зависимость от множителя размера таблицы
     - 1.3.1. Для hFunc1
     - 1.3.2. Для hFunc2
     - 1.3.3. Для hFunc3
2. Анализ собранной информации, сравнение эффективности хеш-функций, выводы о зависимостях эффективности алгоритма от различных параметров
3. Сбор информации о зависимости числа итераций в худшем случае

3.1. Для hFunc1

3.2. Для hFunc2

3.3. Для hFunc3

4. Анализ собранной информации, сравнение экспериментальных значений с теоретическими, выводы о сложности алгоритма вставки в среднем и в худшем случаях

4.2. Технология проведения исследования

В программе есть возможность построения графика и таблицы зависимости числа итераций алгоритма вставки от числа пар, максимальной длины ключа или множителя размера таблицы. Кроме того, подсчитывается время работы алгоритма. Для исследования берутся два показателя – число итераций худшей вставки, то есть такой, которая заняла наибольшее число итераций, и число итераций всех N вставок. Для выбора показателя и параметра для исследования в программе предусмотрены флаги. Пример построения программой графика зависимости числа итераций всех вставок от числа пар представлено на рис. 4.

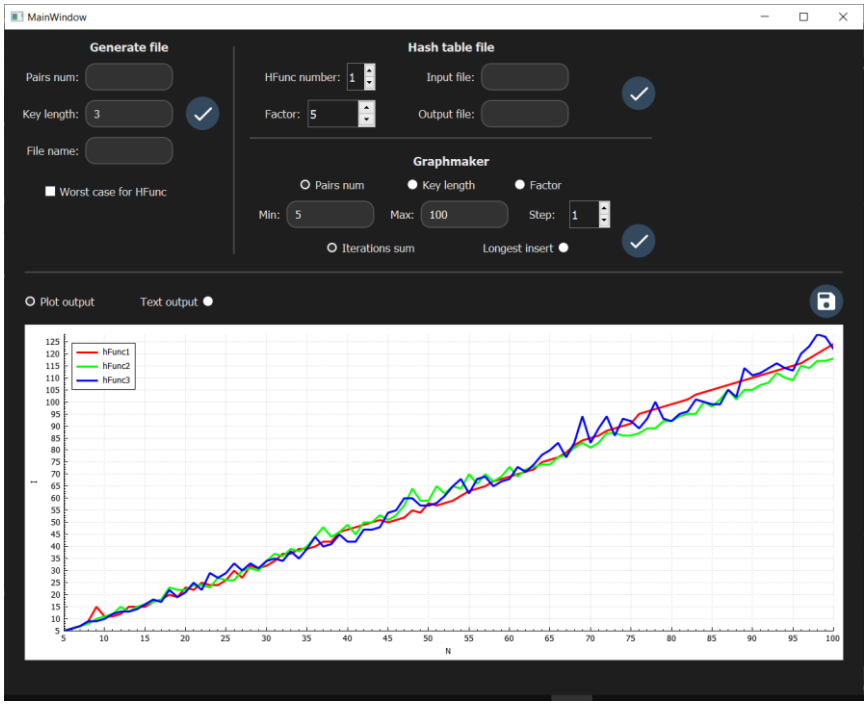

Рисунок 4 – Вывод графика зависимости

Вывод таблицы программой после запуска исследования представлен на рис. 5.

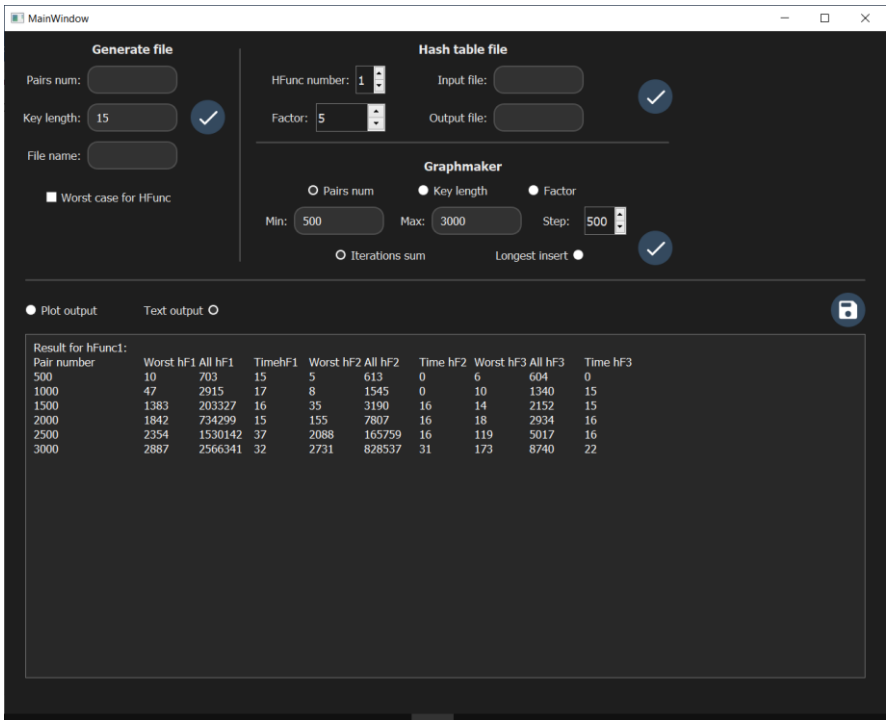

Рисунок 5 – Вывод таблицы зависимости

Таким образом, всю необходимую статистику можно получить из программы, в том числе графическое представление зависимостей, а также параметры затраченного времени.

Обозначения параметров приведены в табл. 13.

Таблица 13 – Основные параметры

Обозначение	Параметр
N	Число пар для вставки
L	Максимальная длина ключа
X	Множитель начальной длины таблицы
H	Начальная длина таблицы, $H = N * X$
S	Шаг изменения переменного параметра
I	Число всех итераций вставки N пар

Окончание таблицы 13

MI	Наибольшее число итераций вставки одной пары

4.3. Исследование зависимостей от максимальной длины ключа

Был проведен ряд тестов, где переменным параметром бралась максимальная длина ключа L. Значения параметров для тестирования №1 приведены в табл. 14.

Таблица 14 – Параметры тестирования №1

Параметр	N	X	L_{min}	L_{max}	S
Значение	20000	10	240	480	10

В результате тестирования программой был построен график зависимости $I(L)$ для каждой из хеш-функций, представленный на рис. 6.

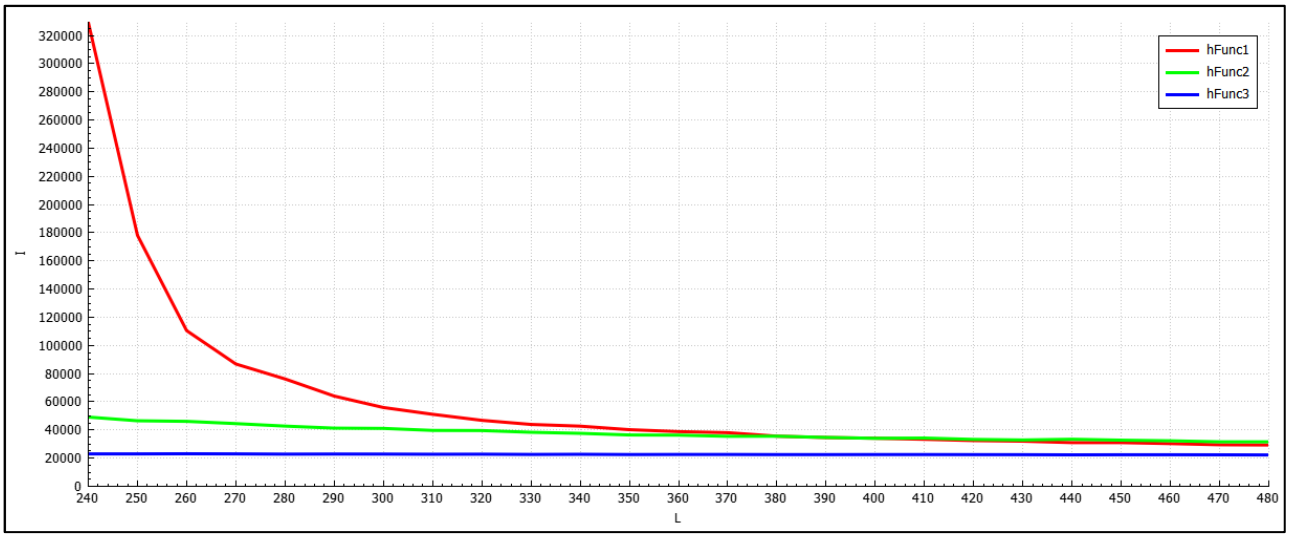

Рисунок 6 – График зависимости $I(L)$ для тестирования №1

Также был построен график зависимости $MI(L)$ для каждой из хеш-функций, представленный на рис. 7.

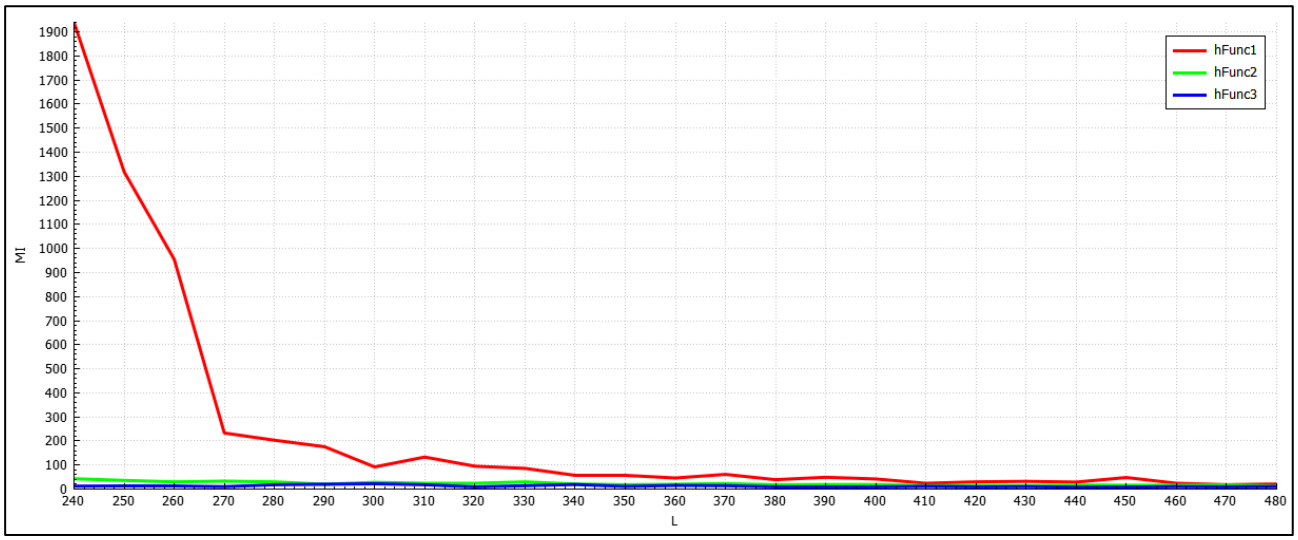

Рисунок 7 – График зависимости $MI(L)$ для тестирования №1

Значения параметров для тестирования №2 представлено в табл. 15.

Таблица 16 – Параметры тестирования №2

Параметр	N	X	L_{min}	L_{max}	S
Значение	5000	5	4	80	2

В результате тестирования программой был построен график зависимости $I(L)$ для каждой из хеш-функций, представленный на рис. 8.

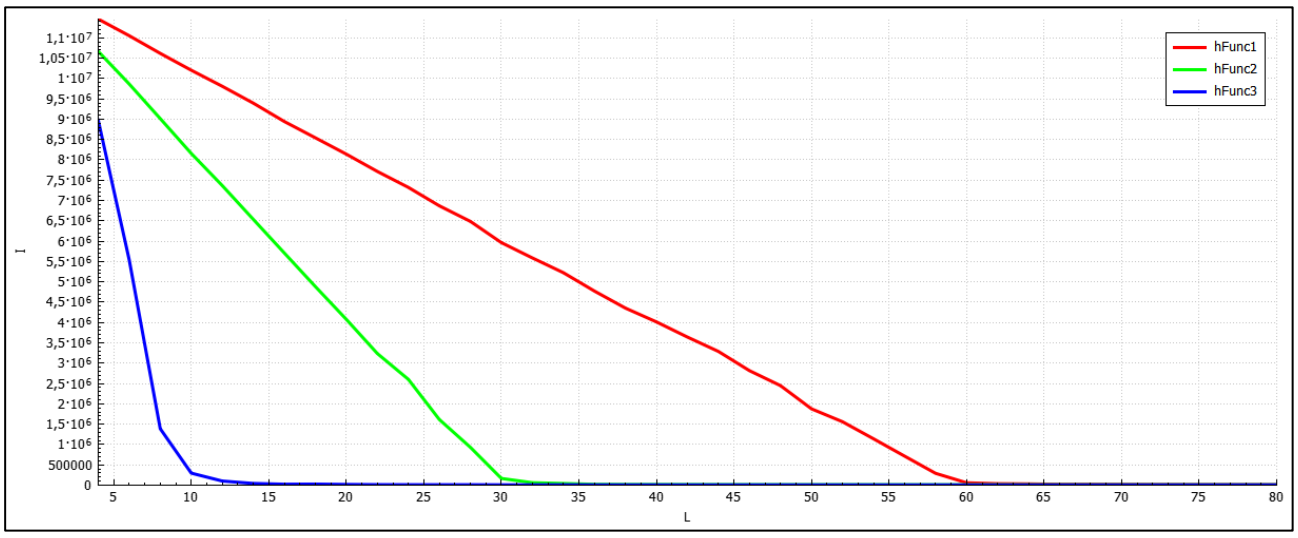

Рисунок 8 – График зависимости $I(L)$ для тестирования №2

Также был построен график зависимости $MI(L)$ для каждой из хеш-функций, представленный на рис. 9.

Рисунок 9 – График зависимости $MI(L)$ для тестирования №2

Из графиков можно увидеть, что эффективность всех хеш-функций зависит от максимальной длины ключей. При малых значениях L возникают худшие случаи вставки, где $MI \approx N$. Однако при хешировании функцией hFunc3 при достаточно небольшом значении L заметен спад числа итераций и исчезновение худших случаев ($MI \ll N$). В случае функций hFunc2 и hFunc1 даже при небольших значениях L создается большое число коллизий, что приводит к многократному возрастанию числа итераций. Хуже всего показывает себя хеш-функция hFunc1, для эффективной работы которой требуется наибольшая длина ключей.

В целом можно сделать вывод, что для каждой хеш-функции и числа пар N существует константа C такая, что при $L < C$ числа итераций I и MI растут обратно пропорционально L, причем MI растет до значения N, а в случае $L > C$ числа итераций не имеют четкой зависимости от длины ключа.

4.4. Исследование зависимостей от множителя размера таблицы

Был проведен ряд тестов, где переменным брался множитель размера таблицы X. Значение параметров N и L подбиралось на основе проведенного ранее исследования так, чтобы длина ключа не вызывала большой разницы в

эффективности работы хеш-функций. Значения параметров для тестирования №3 приведены в табл. 17.

Таблица 17 – Параметры тестирования №3

Параметр	N	L	X_{min}	X_{max}	S
Значение	10000	500	1	15	1

В результате тестирования программой был построен график зависимости $I(X)$ для каждой из хеш-функций, представленный на рис. 10.

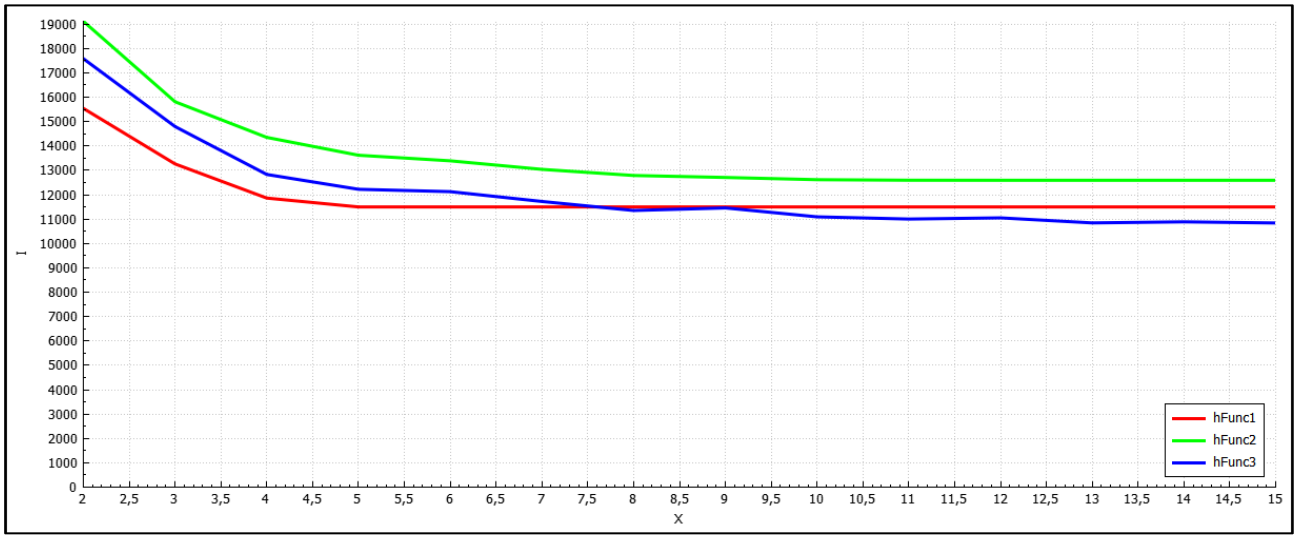

Рисунок 10 – График зависимости $I(X)$ для тестирования №3

Также был построен график зависимости $MI(X)$ для каждой из хеш-функций, представленный на рис. 11.

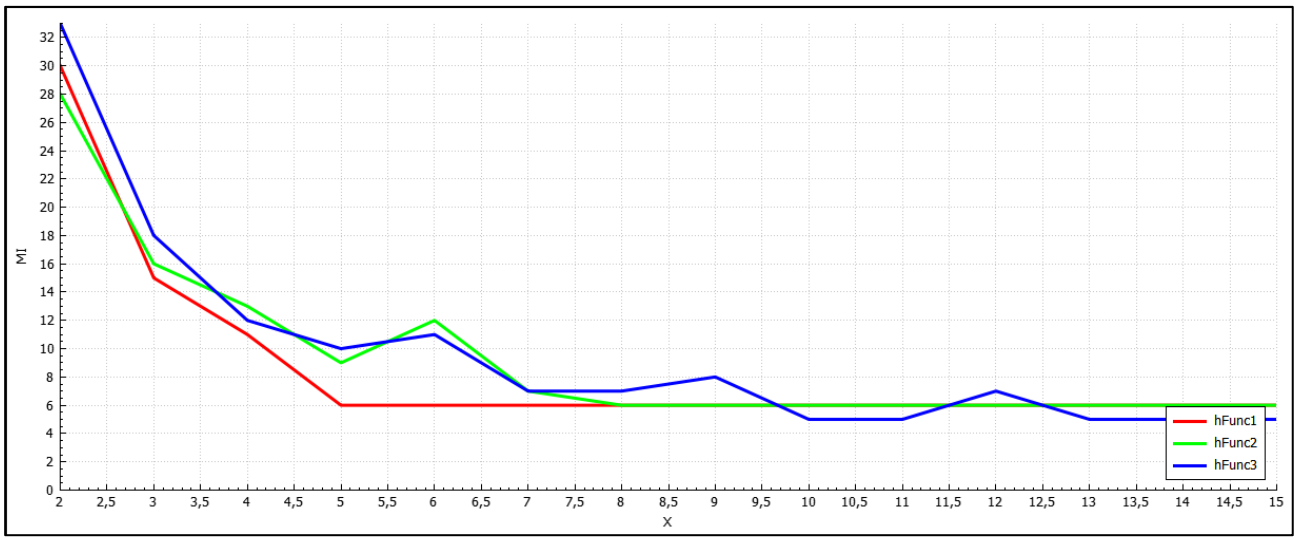

Рисунок 11 – График зависимости $MI(X)$ для тестирования №3

Значения параметров для тестирования №4 представлено в табл. 18.

Таблица 18 – Параметры тестирования №4

Параметр	N	L	X_{min}	X_{max}	S
Значение	10000	150	2	15	1

В результате тестирования программой был построен график зависимости $I(X)$ для каждой из хеш-функций, представленный на рис. 12.

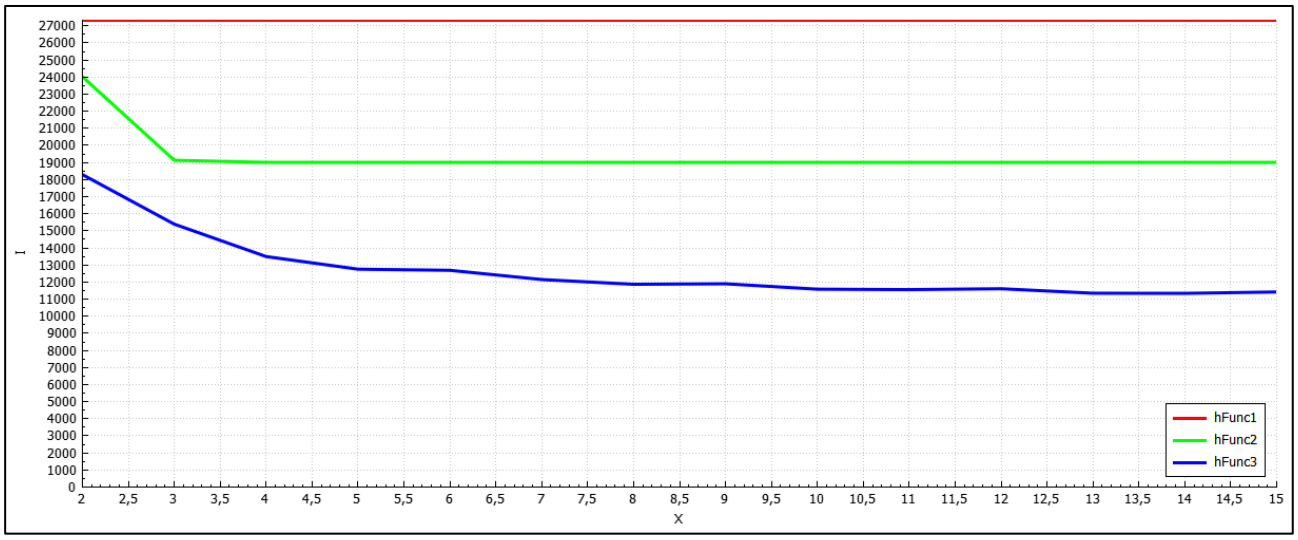

Рисунок 12 – График зависимости $I(X)$ для тестирования №4

Также был построен график зависимости $MI(X)$ для каждой из хеш-функций, представленный на рис. 13.

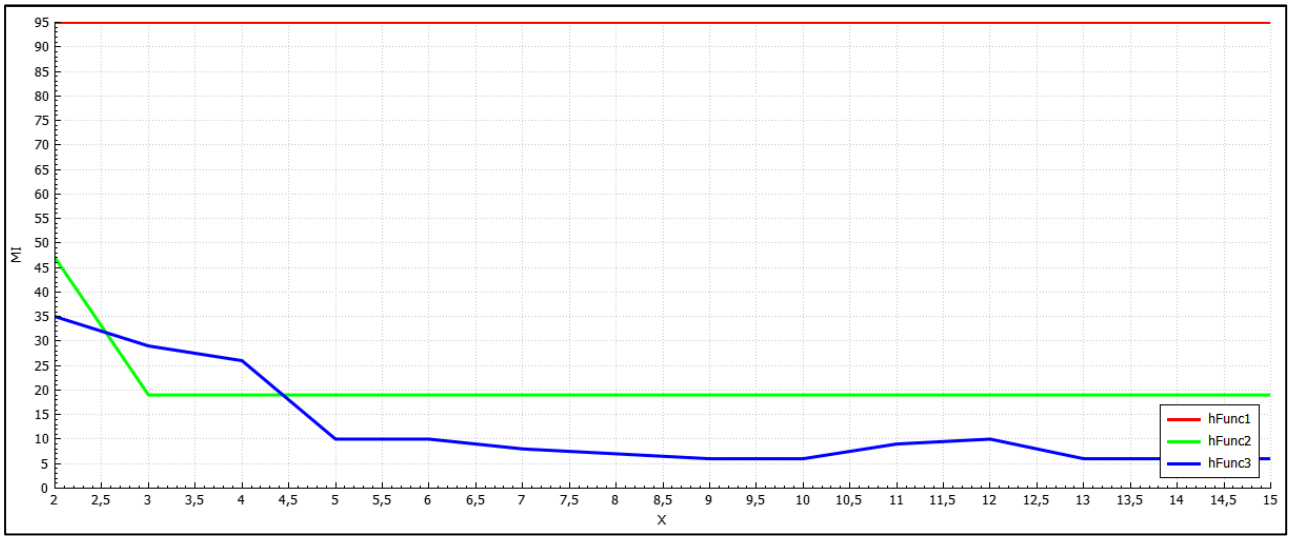

Рисунок 13 – График зависимости $MI(X)$ для тестирования №4

Из графиков можно увидеть, что зависимости $I(X), MI(X)$ ведут себя по-разному в зависимости от L. При достаточно большом значении L, при котором значения $I(L)$ для каждой хеш-функции сравнимы друг с другом, при любом алгоритме хеширования наблюдается падение числа итераций в диапазоне $2 \leq X \leq 6$. При хешировании hFunc3 падение продолжается с увеличением X.

При небольшом значении L, при котором значения $I(L)$ для каждой хеш-функции отличаются, но не в разы, наблюдается другая картина: при хешировании hFunc1 параметры I и MI не зависят от X, при hFunc2 только при $X = 2$ наблюдается изменение параметров, а при хешировании hFunc3 все так же наблюдается падение числа итераций на всем диапазоне.

В результате можно сделать вывод, что малые значения X снижают эффективность хеширования и алгоритма вставки, а большие значения X дают преимущество только для определенных хеш-функций.

4.5. Исследование зависимостей от числа пар

Был проведен ряд тестов, где переменным бралось число пар N. Значение параметра X подбиралось на основе проведенного ранее исследования так, чтобы размер хеш-таблицы не вызывал большой разницы в эффективности работы хеш-функций. Значения параметров для тестирования №5 приведены в табл. 19.

Таблица 19 – Параметры тестирования №5

Параметр	L	X	N_{min}	N_{max}	S
Значение	200	10	1000	10000	200

В результате тестирования программой был построен график зависимости $I(N)$ для каждой из хеш-функций, представленный на рис. 14.

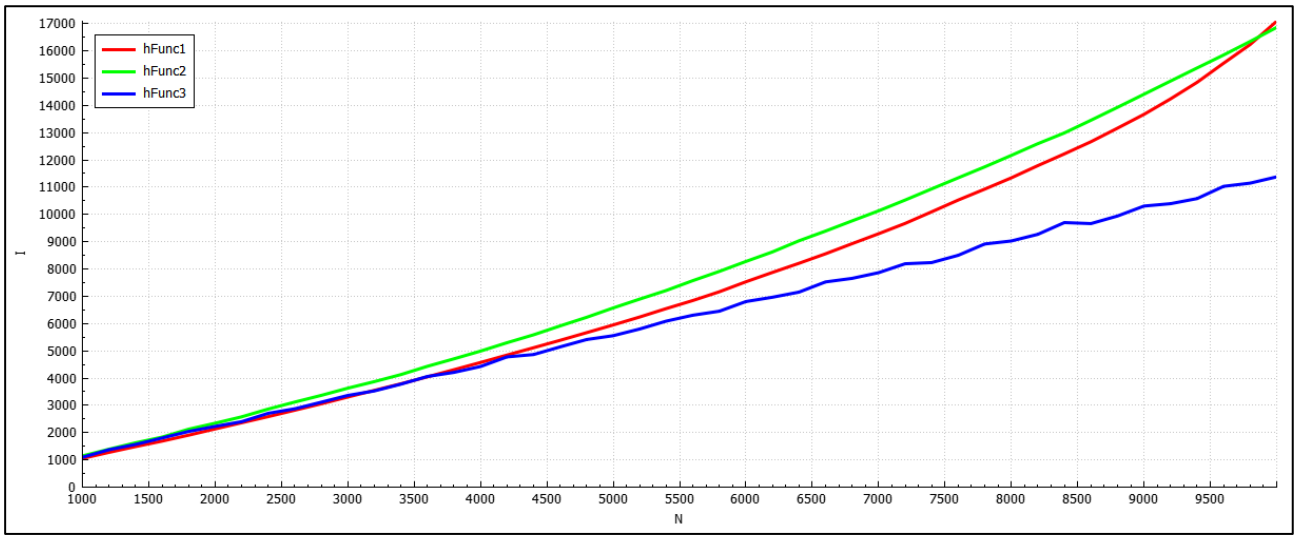

Рисунок 14 – График зависимости $I(N)$ для тестирования №5

Также был построен график зависимости $MI(N)$ для каждой из хеш-функций, представленный на рис. 15.

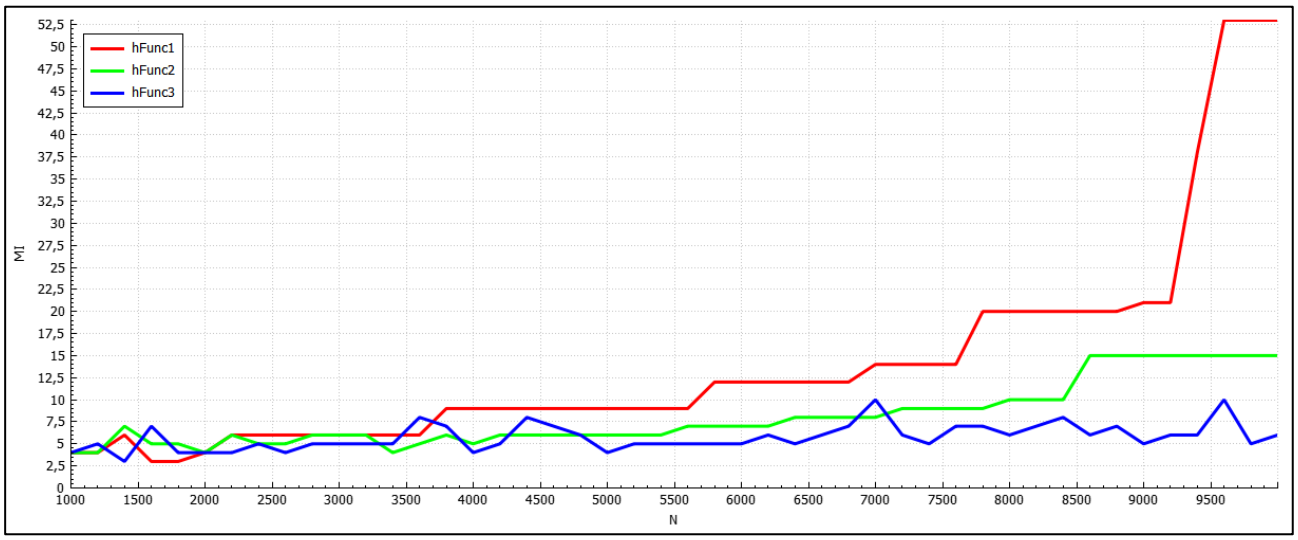

Рисунок 15 – График зависимости $MI(N)$ для тестирования №5

Значения параметров для тестирования №6 приведены в табл. 20.

Таблица 20 – Параметры тестирования №6

Параметр	L	X	N_{min}	N_{max}	S
Значение	140	10	1000	10000	200

В результате тестирования программой был построен график зависимости $I(N)$ для каждой из хеш-функций, представленный на рис. 16.

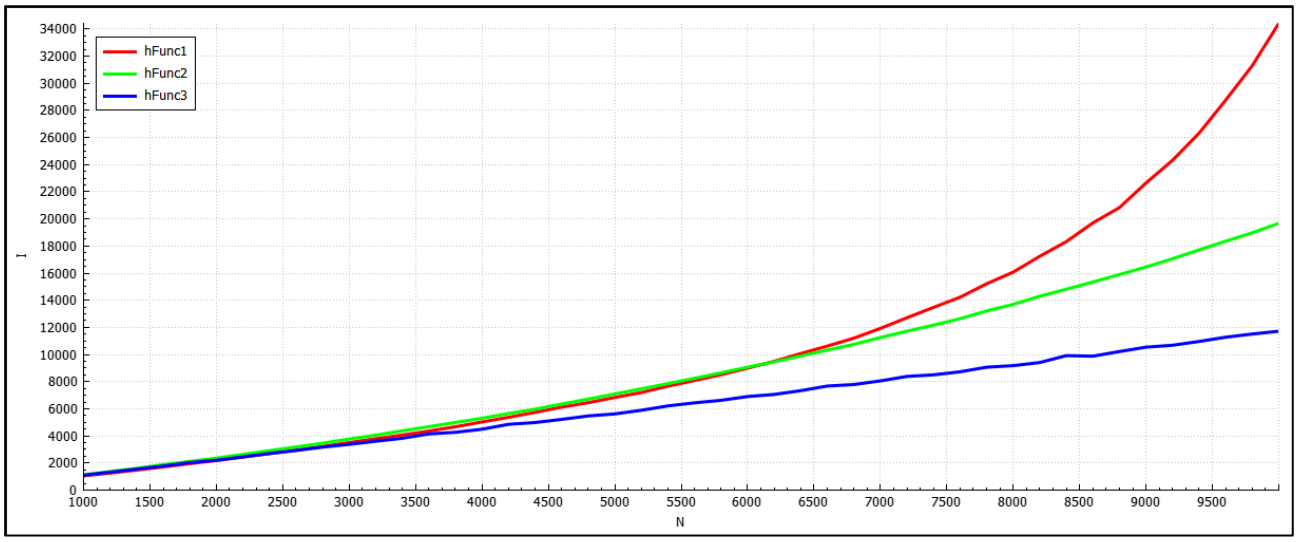

Рисунок 16 – График зависимости $I(N)$ для тестирования №6

Также был построен график зависимости $MI(N)$ для каждой из хеш-функций, представленный на рис. 17.

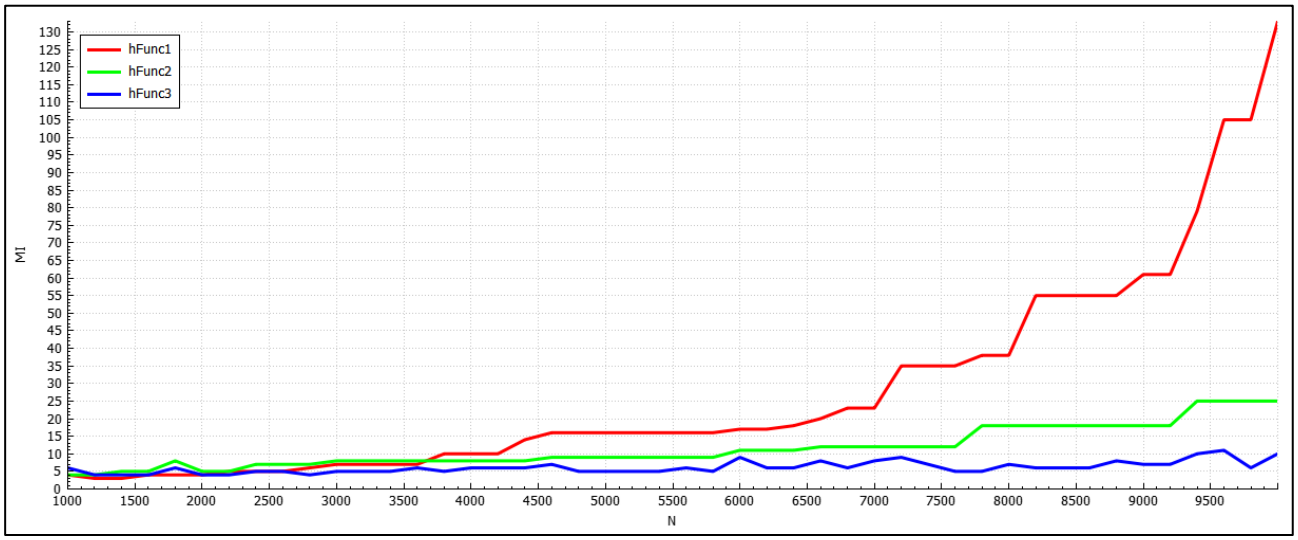

Рисунок 17 – График зависимости $MI(N)$ для тестирования №6

Из тестирований №5 и №6 видно, что большую роль в зависимости $I(N)$ и $MI(N)$ играет максимальная длина ключа L, так как она ограничивает диапазон результатов некоторых хеш-функций, в частности, hFunc1. При достаточно большом L не возникает большого количества коллизий, ввиду чего зависимость $I(N)$ линейна, что видно в тестировании №5. Однако для каждой хеш-функции и каждого значения L есть такое значение C, что при $N > C$ зависимость перестает быть линейной ввиду множественных коллизий.

Значения параметров для тестирования №7 приведены в табл. 21.

Таблица 21 – Параметры тестирования №7

Параметр	L	X	N_{min}	N_{max}	S
Значение	70	10	1000	10000	200

В результате тестирования программой был построен график зависимости $I(N)$ для hFunc2 и hFunc3, представленный на рис. 18.

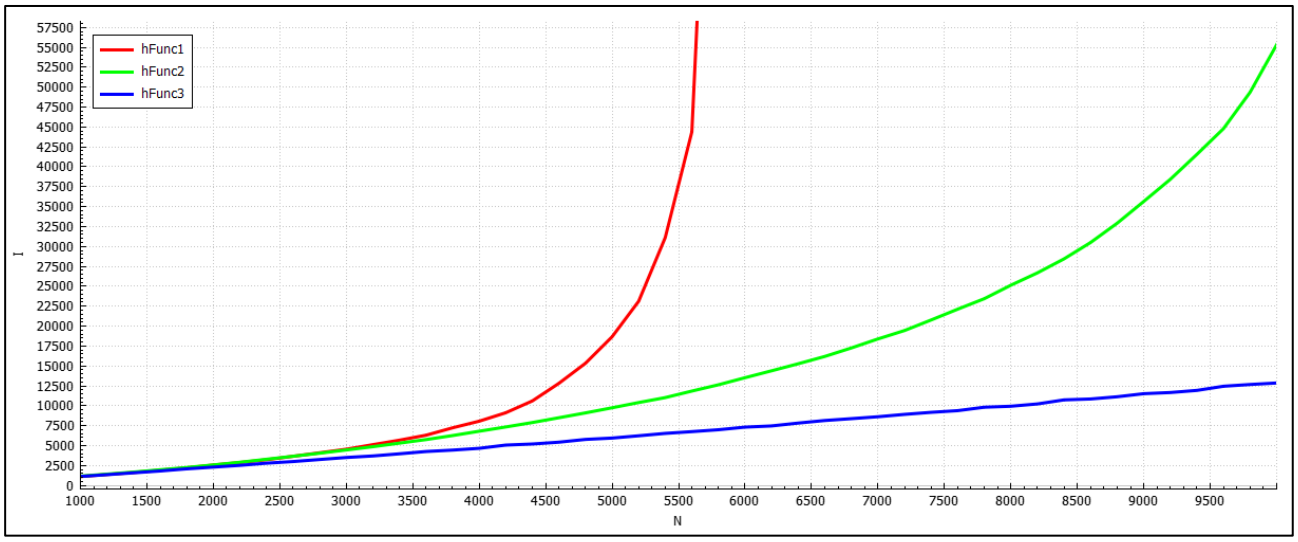

Рисунок 18 – График зависимости $I(N)$ для тестирования №7

Также был построен график зависимости $MI(N)$ для каждой из хеш-функций, представленный на рис. 19.

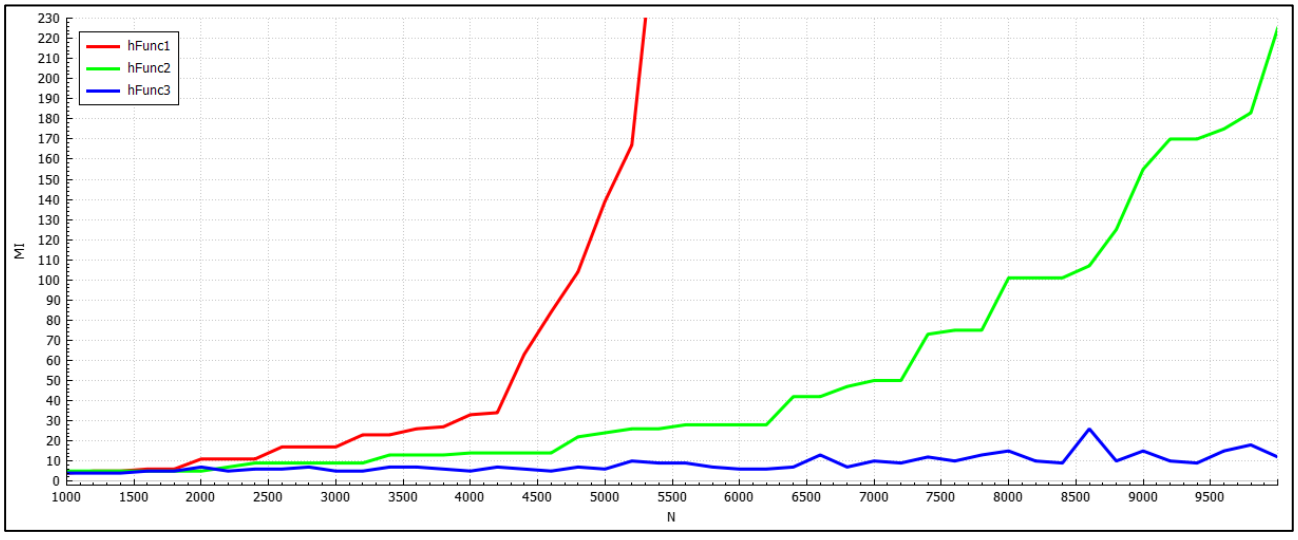

Рисунок 19 – График зависимости $MI(N)$ для тестирования №7

Как видно из графиков, зависимость $I(N)$ для хеш-функции hFunc2 нелинейная, а зависимость максимального числа итераций для одной вставки $MI(N)$ с некоторого момента начинает значительно возрастать. Стоит отметить, что в тестированиях №5-7 зависимость $I(N)$ хеш-функции hFunc3 можно охарактеризовать как линейную.

Возможно подобрать такое значение L, при котором хеш-функция hFunc3 также станет менее эффективной. Значения параметров для тестирования №8 приведены в табл. 22.

Таблица 22 – Параметры тестирования №8

Параметр	L	X	N_{min}	N_{max}	S
Значение	10	10	1000	10000	200

В результате тестирования программой был построен график зависимости $I(N)$ для хеш-функций, представленный на рис. 20.

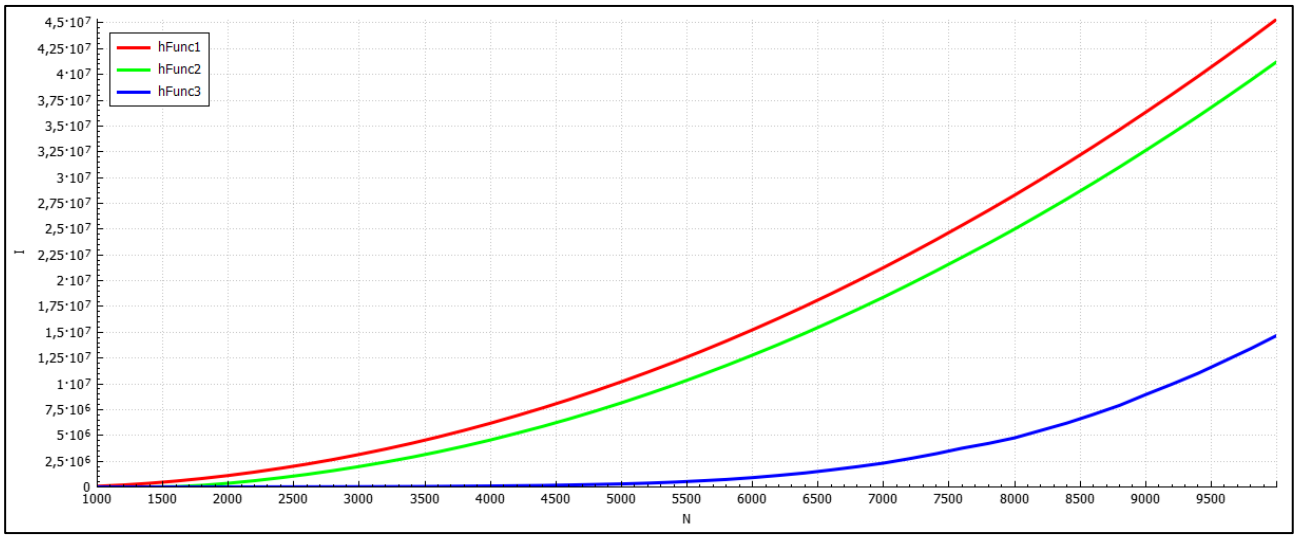

Рисунок 20 – График зависимости $I(N)$ для тестирования №8

Из графика видно, что зависимость $I(N)$ для всех трех хеш-функций нелинейная, однако число итераций для хеш-функции hFunc3 значительно меньшее, чем для других хеш-функций.

Также был построен график зависимости $MI(N)$ для каждой из хеш-функций, представленный на рис. 21.

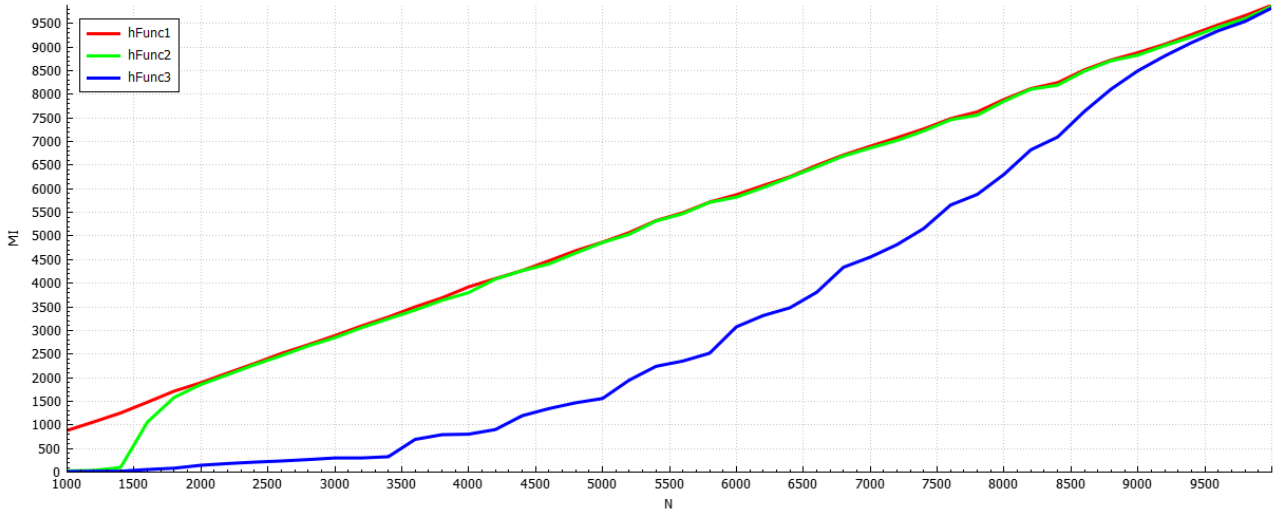

Рисунок 21 – График зависимости $MI(N)$ для тестирования №8

Как видно из графика, для хеш-функций hFunc1 и hFunc2 уже при $N > 2000$ возникает худший случай, то есть $MI \approx N$. Зависимость $MI(N)$ для hFunc3 также растет, но худший случай можно заметить лишь в конце исследуемого диапазона, при $N > 9800$.

4.6. Вывод об эффективности хеш-функций

Наиболее эффективной хеш-функция из исследуемых оказалась хеш-функция hFunc3. В большинстве тестов функция обеспечивала линейную зависимость $I(N)$, а также $MI \ll N$, что гарантирует сложность алгоритма вставки одного элемента $O(1)$, а сложность алгоритма вставки N элементов $O(N)$. Однако в критических ситуациях (при малом размере хеш-таблицы, либо малой длине ключей, но при этом большом значении N) все три хеш-функции снижают эффективность алгоритма из-за ограничения диапазона выходного значения при хешировании. Таким образом, даже при случайных данных могут возникать случаи, когда существует такое число C, что при $N > C$ значение $MI \approx N$, а зависимость $I(N)$ нелинейная. В таких случаях нельзя гарантировать сложность алгоритма вставки одного элемента $O(1)$.

4.7. Исследование худшего случая вставки

Для каждой хеш-функции можно сгенерировать такие входные данные, что каждый ключ после хеширования будет давать одинаковое число, из-за чего при каждой вставке будет иметь место худший случай.

Значения параметров для тестирования №9 приведены в табл. 23.

Таблица 23 – Параметры тестирования №8

Параметр	L	X	N_{min}	N_{max}	S
Значение	400	10	500	5000	100

В результате тестирования программой был построен график зависимости $I(N)$ для каждой из хеш-функций. Для наглядности, графики имеют разную толщину линий. Графики представлены на рис. 22.

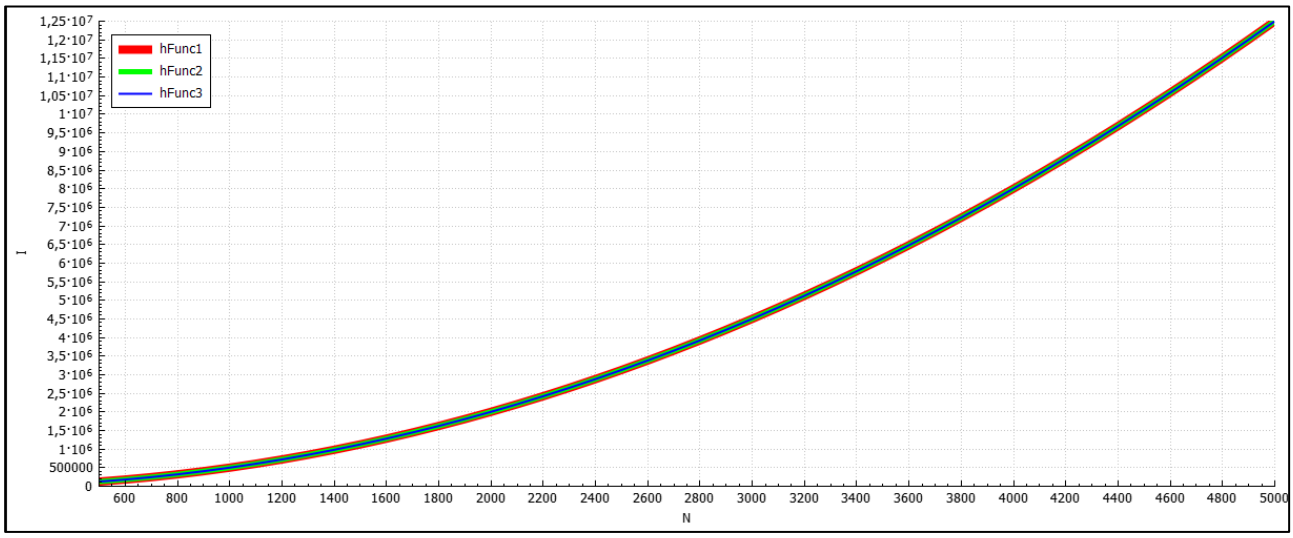

Рисунок 22 – График зависимости $I(N)$ для тестирования №9

Также был построен график зависимости $MI(N)$ для каждой из хеш-функций, представленный на рис. 23.

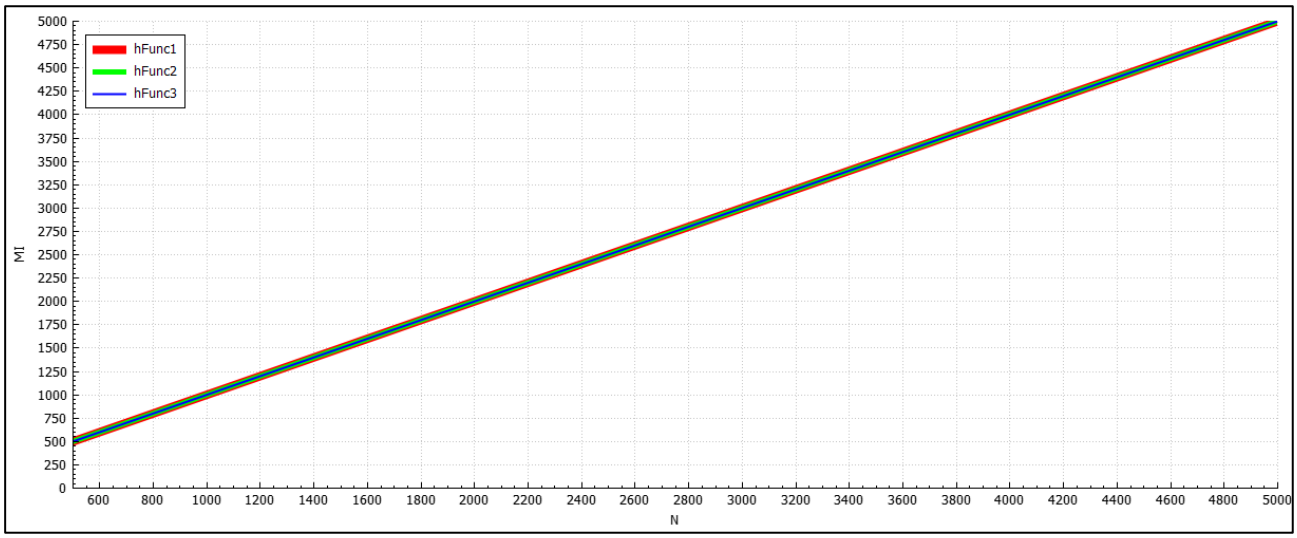

Рисунок 23 – График зависимости $MI(N)$ для тестирования №9

Из рис. 23 видно, что зависимость $MI(N)$ линейная для всех трех хеш-функций, причем $MI = N$, что и соответствует худшему случаю вставки. Сложность алгоритма вставки одного в худшем элемента $O(N)$. Зависимость $I(N)$ всех хеш-функций одинаковая и нелинейная. Чтобы точнее оценить ее, были взяты значения I при некоторых N, указанные в табл. 24. Так как при каждой вставке ожидается худший случай, то теоретически число итераций равно сумме целых чисел от 1 до N.

Таблица 24 – Соответствие $I(N)$ для тестирования №9

N	I	$\sum_{i=1}^{N} i$
500	125250	125250
1500	1125750	1125750
3000	4501500	4501500
4500	10127250	10127250

Из таблицы видно, что число итераций действительно равно сумме всех чисел от 1 до N, а значит для алгоритма вставки N элементов сложность $O(N^2)$.

4.8. Выводы об исследовании алгоритма

Исследование показало, что эффективность алгоритма вставки зависит от хеш-функции, а также от таких параметров, как размер хеш-таблицы и максимальная длина ключа. Теоретически в среднем случае сложность алгоритма вставки одного элемента $O(1)$. Среди исследуемых хеш-функций самой устойчивой к параметрам, то есть чаще всего обеспечивающей условия среднего случая, оказалась хеш-функция с названием hFunc3, а самой неустойчивой, то есть реже всего обеспечивающей условия среднего случая, - hFunc1. Выбор наиболее эффективной хеш-функции позволяет обеспечить быструю работу хеш-таблицы с большим объемами различных данных, не занимая при этом большое количество памяти. Так, для hFunc3 требуется меньший множитель размера хеш-таблицы X, чтобы при прочих равных параметрах обеспечить сложность $O(1)$ для вставки одного элемента, что экономит память.

Исследование показало, что в худшем случае для любых хеш-функций сложность алгоритма вставки одного элемента $O(N)$, так как $MI = N$ при любых N. В то же время, сложность алгоритма вставки N элементов $O(N^2)$, а точное число итераций высчитывается по формуле $I = \sum_{i=1}^{N} i$.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была разработана программа, которая обладает следующей функциональностью: создание файла, содержащего входные данные для хеш-таблицы, создание хеш-таблицы по заданному файлу, хеш-функции и размеру хеш-таблицы; построение графиков зависимости числа итераций от нескольких параметров хеш-таблицы или входных данных для каждой хеш-функции. С помощью программы было проведено исследование среднего и худшего случаев алгоритма вставки в хеш-таблицу с различными хеш-функциями. В ходе исследования была исследована зависимость эффективности алгоритма от различных параметров хеш-таблицы, а также от выбора хеш-функции. В результате было выявлено, что на эффективность значительно влияет как выбор алгоритма хеширования, так и размер таблицы, а также максимальная длина ключа. При этом, в худшем случае сложность алгоритма вставки $O(N)$, а в среднем случае $O(1)$. При выборе неэффективной хеш-функции при широком диапазоне параметров может наблюдаться худший случай, а при выборе эффективной хеш-функции уменьшается количество коллизий и реже возникают худшие случаи вставки в хеш-таблицу.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Bjarne Stroustrup. A Tour of C++. M.: Addison-Wesley, 2018. 217 с.
2. Макс Шлее. Qt5.10. Профессиональное программирование на C++. M.: BHV-СПб, 2018, 513 с.
3. Перевод и дополнение документации QT // CrossPlatform.RU. URL: http://doc.crossplatform.ru/
4. Qt Documentation // Qt. URL: https://doc.qt.io/qt-5/index.html
5. Documentation // QCustomPlot. URL: https://www.qcustomplot.com/index.php/support/documentation
6. Хеш-таблицы // AlgoList. URL: http://algolist.ru/ds/s_has.php
7. Примеры хеш-функций // Poznayka. URL: https://poznayka.org/s97484t1.html

ПРИЛОЖЕНИЕ А.

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAIN.C

```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД ПРОГРАММЫ. PAIR.H

```
#ifndef PAIR_H
#define PAIR_H
#include "libraries.h"

// Class assignment:        Pair of 2 elements of any type for HASH TABLE
// Features:                Keeps info about being DELETED or being ACTIVE
template <typename S, typename T>
class Pair
{
public:
    Pair();
    Pair(S,T);
    Pair(Pair &);
    ~Pair();
    const Pair & operator=(const Pair &other);

    string toString();

    S getFirst();
    T getSecond();
    void setFirst(S);
    void setSecond(T);
    void setDeleted();

    int isActive();
    int wasDeleted();
private:
    S *f;           // FIRST element
    T *s;           // SECOND element
    bool active;    // ACTIVE state
    bool deleted;   // DELETED state
};

// Function assignment:     Empty pair constructor
// Features:                Sets ACTIVE and DELETED to false
template <typename S, typename T>
Pair<S,T>::Pair()
{
    f = NULL;
    s = NULL;
    active = false;
    deleted = false;
}

// Function assignment:     Pair constructor by two elements
```

```
// Features:                Sets ACTIVE to true and DELETED to false
template <typename S, typename T>
Pair<S,T>::Pair(S x, T y)
{
    f = new S;  *f = x;
    s = new T;  *s = y;
    active = true;
    deleted = false;
}

// Function assignment:     Pair constructor by another pair
// Features:                1. Copy elements from the other pair
//                          2. Copy ACTIVE and DELETED states
template <typename S, typename T>
Pair<S,T>::Pair(Pair &other)
{
    f = NULL; s = NULL;
    if (other.f != NULL)
        f = new S(*other.f);
    if (other.s != NULL)
        s = new T(*other.s);
    active = other.active;
    deleted = other.deleted;
}

// Function assignment:     Returns pair's FIRST element value
// Features:                Read only
template <typename S, typename T>
S Pair<S,T>::getFirst()
{
    if (f != NULL)
        return *f;
    else
        return NULL;
}

// Function assignment:     Returns pair's SECOND element value
// Features:                Read only
template <typename S, typename T>
T Pair<S,T>::getSecond()
{
    if (s != NULL)
        return *s;
    else
        return NULL;
}

// Function assignment:     Sets FIRST pair element
```

```
// Features:                1. Allocates memory if necessary
//                          2. Sets ACTIVE to true and DELETED to false
template <typename S, typename T>
void Pair<S,T>::setFirst(S x)
{
    if (f==NULL)
        f = new S;
    *f = x;
    active = true;
    deleted = false;
}

// Function assignment:     Sets SECOND pair element
// Features:                1. Allocates memory if necessary
//                          2. Sets ACTIVE to true and DELETED to false
template <typename S, typename T>
void Pair<S,T>::setSecond(T y)
{
    if (s == NULL)
        s = new T;
    *s = y;
    active = true;
    deleted = false;
}

// Function assignment:     Creates string in the format "(FIRST, SECOND)"
// Features:                1. If element is empty, writes "NULL" in its place
//                          2. If pair is empty, writes "NULL"
template <typename S, typename T>
string Pair<S,T>::toString()
{
    stringstream ss;
    if (active)
    {
        ss << "(";
        if (f == NULL)
            ss << "NULL";
        else
            ss << (*f);
        ss<<" ";
        if (s == NULL)
            ss << "NULL";
        else
            ss << (*s);
        ss << ")";
    }
    else
    {
```

```
        ss << "NULL";
    }
    return ss.str();
}

// Function assignment:     Returns true if pair is ACTIVE
template <typename S, typename T>
int Pair<S,T>::isActive()
{
    return active;
}

// Function assignment:     Returns true if pair is DELETED
template <typename S, typename T>
int Pair<S,T>::wasDeleted()
{
    return deleted;
}

// Function assignment:     Pair destructor
// Features:                1. Deletes elements
//                          2. Set ACTIVE and DELETED to false
template <typename S, typename T>
Pair<S,T>::~Pair()
{
    if (f != NULL)
        delete f;
    if (s != NULL)
        delete s;
    f = NULL;
    s = NULL;
    deleted = false;
    active = false;
}

// Function assignment:     Set DELETED to true
template <typename S, typename T>
void Pair<S,T>::setDeleted()
{
    deleted = true;
}

// Function assignment:     Copy another pair by operator =
// Features:                1. Deletes old elements if necessary
//                          2. Copy elements from another pair
//                          3. Copy ACTIVE and DELETED states
template <typename S, typename T>
const Pair<S,T> & Pair<S,T>::operator=(const Pair<S,T> &other)
```

```
{
    if(this != &other)
    {
        if(f != NULL)
        delete f;
        if(s != NULL)
        delete s;
        f = NULL; s = NULL;
        if (other.f != NULL)
        f = new S(*other.f);
        if (other.s != NULL)
        s = new T(*other.s);
        deleted = other.deleted;
        active = other.active;
    }
    return *this;
}

#endif // PAIR_H
```

ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ КОД ПРОГРАММЫ. CVECTOR.H

```
#ifndef PAIRLIST_H
#define PAIRLIST_H
#include "pair.h"

// Class assignment:        Vector of elements any type (dynamic) for HASH TABLE
// Features:                Keeps info about it's CAPACITY
template <typename T>
class CVector
{
public:
    CVector();
    CVector(unsigned int nsize);
    unsigned int getCapacity();
    int length();
    int resize(unsigned int nsize);
    T & operator[](unsigned int index);
    CVector<T> & operator=(const CVector<T> &);
private:
    T *tail;                // Dynamic ARRAY of elements
    unsigned int capacity;  // Vector's CAPACITY
};

// Function assignment:     Empty vector constructor
// Features:                   Allocates memory for one ARRAY element (updates
CAPACITY)
template <typename T>
CVector<T>::CVector()
{
    tail = new T[1];
    capacity = 1;
};

// Function assignment:     Empty vector with fixed CAPACITY constructor
// Features:                Allocates memory for ARRAY (updates CAPACITY)
template <typename T>
CVector<T>::CVector(unsigned int nsize)
{
    capacity = nsize;
    tail = new T[capacity];
}

// Function assignment:     Returns vector's CAPACITY
template <typename T>
unsigned int CVector<T>::getCapacity()
{
```

```
    return capacity;
}

// Function assignment:     Returns vector's CAPACITY
template <typename T>
int CVector<T>::length()
{
    return capacity;
}

// Function assignment:     Reallocates memory for ARRAY (updates CAPACITY)
template <typename T>
int CVector<T>::resize(unsigned int nsize)
{
    if (nsize <= capacity)
        return 1;
    T *temp = new T[nsize];
    for (unsigned int i=0; i<capacity; i++)
    {
        temp[i] = tail[i];
    }
    capacity = nsize;
    delete [] tail;
    tail = temp;
    return 0;
}

template <typename T>
T& CVector<T>::operator[](unsigned int index)
{
    return tail[index];
}

template<class T>
CVector<T> & CVector<T>::operator = (const CVector<T> & v)
{
    delete[] tail;
    capacity = v.capacity;
    tail = new T[capacity];
    for (unsigned int i = 0; i < capacity; i++)
        tail[i] = v.tail[i];
    return *this;
}

#endif // PAIRLIST_H
```

ПРИЛОЖЕНИЕ Г

ИСХОДНЫЙ КОД ПРОГРАММЫ. HASHTABLE.H

```
#ifndef HASHTABLE_H
#define HASHTABLE_H
#include "cvector.h"
#define SIZE 256

template <typename S, typename T>
class HashTable
{
public:
    HashTable();
    HashTable(unsigned int size);

    void setSize(int size);
    unsigned int getSize();
    void setStep(bool s);
    unsigned int getItemCounter();

    Pair<S,T> & operator[](unsigned int index);
    unsigned int set(S x, T y, unsigned int index);
    Pair<S,T> get(S key, unsigned int index);
    unsigned int remove(S key, unsigned int index);
    void clear();

    string getString(S key, unsigned int index);
    string print();
    string printStep(unsigned int index);
private:
    CVector< Pair<S,T> > table;
    unsigned int itemCounter;
    bool step;
};

template <typename S, typename T>
HashTable<S,T>::HashTable()
{
    itemCounter = 0;
    setSize(SIZE);
}

template <typename S, typename T>
HashTable<S,T>::HashTable(unsigned int size)
{
    itemCounter = 0;
    setSize(size);
}
```

```
template <typename S, typename T>
void HashTable<S,T>::setSize(int size)
{
    table.resize(size);
}

template <typename S, typename T>
void HashTable<S,T>::setStep(bool s)
{
    step = s;
}

template <typename S, typename T>
unsigned int HashTable<S,T>::getSize()
{
    return table.getCapacity();
}

template <typename S, typename T>
unsigned int HashTable<S,T>::getItemCounter()
{
    return itemCounter;
}

template <typename S, typename T>
unsigned int HashTable<S,T>::set(S x, T y, unsigned int index)
{
    unsigned int counter = 1;
    if (index + 5 >= table.getCapacity())
        table.resize((index + 5) * 2);
    while (table[index].isActive())
    {
        index++;
        counter++;
        if (index + 5 >= table.getCapacity())
            table.resize((index + 5) * 2);
    }
    table[index].setFirst(x);
    table[index].setSecond(y);
    itemCounter++;
    return counter;
}

template <typename S, typename T>
Pair<S,T> HashTable<S,T>::get(S key, unsigned int index)
{
    while (table[index].isActive())
```

```
    {
        if (table[index].wasDeleted())
        {
            index++;
            continue;
        }
        if (table[index].getFirst() != key)
            index++;
        else
            break;
    }
    return table[index];
}

template <typename S, typename T>
unsigned int HashTable<S,T>::remove(S key, unsigned int index)
{
    unsigned int counter = 1;
    while (table[index].isActive())
    {
        printStep(index);
        if (table[index].wasDeleted())
        {
            index++;
            counter++;
            continue;
        }
        if (table[index].getFirst() != key)
        {
            index++;
            counter++;
        }
        else
            break;
    }
    if (table[index].isActive())
    {
        table[index].~Pair();
        table[index].setDeleted();
        itemCounter -= 1;
        return counter;
    }
    else
        return 0;

}

template <typename S, typename T>
```

```
void HashTable<S,T>::clear()
{
    for (unsigned int i=0; i<table.getCapacity(); i++)
    {
        table[i].~Pair();
    }
    itemCounter = 0;
}

template <typename S, typename T>
string HashTable<S,T>::getString(S key, unsigned int index)
{
    string output;
    while (table[index].isActive())
    {
        if (table[index].getFirst() != key)
            index++;
        else
            break;
    }
    if (table[index].isActive())
        output = table[index].toString();
    else
        output = "No key in table";
    return output;
}

template <typename S, typename T>
string HashTable<S,T>::print()
{
    string output;
    for (unsigned int i=0; i<table.getCapacity(); i++)
    {
        if (table[i].isActive() && !table[i].wasDeleted())
        {
            output += to_string(i);
            output += "\t";
            output += table[i].toString();
            output += "\n";
        }
    }
    return output;
}

template <typename S, typename T>
string HashTable<S,T>::printStep(unsigned int index)
{
    string output;
```

```
    for (unsigned int i=0; i<table.getCapacity(); i++)
    {
        if (table[i].isActive() && !table[i].wasDeleted())
        {
            if (i == index)
                output += "{";
            output += to_string(i);
            output += "\t";
            output += table[i].toString();
            if (i == index)
                output += "}";
            output += "\n";
        }
        if (table[i].wasDeleted() && i == index)
        {
            output += "Passing deleted element\n";
        }
    }
    QMessageBox out;
    out.setStandardButtons(QMessageBox::Ok | QMessageBox::Cancel);
    out.setText(QString::fromStdString(output));
    if (out.exec() == QMessageBox::Cancel)
              return output;
}

template <typename S, typename T>
Pair<S,T> & HashTable<S,T>::operator[](unsigned int index)
{
    return table[index];
}

#endif // HASHTABLE_H
```

ПРИЛОЖЕНИЕ Д

ИСХОДНЫЙ КОД ПРОГРАММЫ. STRSTRWORKER.H

```
#ifndef STRSTRWORKER_H
#define STRSTRWORKER_H
#include "hashtable.h"

struct hCreateInfo
{
    unsigned int hSize;
    unsigned itemCounter;
    unsigned int sumIterNum;
    unsigned int maxIterNum;
    bool error;
};

class StrStrWorker
{
public:
    StrStrWorker();
    QString getFromFile(QString fileName);
    hCreateInfo createHashTable(QString input, int hFuncNum, int factor, QString
fileName);
    QString deleteElem(QString input, bool step);
    void saveStrToFile(QString output, QString fileName);
    int generateFile(int size, int maxLength, QString fileName, bool worstCase,
int hFuncNum);
private:
    HashTable <string, string> hTable;
    unsigned int hFunc1(string key, unsigned int size);
    unsigned int hFunc2(string key, unsigned int size);
    unsigned int hFunc3(string key, unsigned int size);
    hCreateInfo info;
    int iter;
};

#endif // STRSTRWORKER_H
```

ПРИЛОЖЕНИЕ Е

ИСХОДНЫЙ КОД ПРОГРАММЫ. STRSTRWORKER.CPP

```
#include "strstrworker.h"

StrStrWorker::StrStrWorker()
{
    iter = 0;
}
int StrStrWorker::generateFile(int size, int maxLength, QString fileName, bool
worstCase, int hFuncNum)
{
    string file = fileName.toStdString();
    ofstream out(file);
    QMessageBox message;
    if (!out)
        return BAD_FILE;
    if (size <= 0 || size > SIZE_LIMIT)
        return BAD_SIZE;
    if (maxLength <= 0 || (maxLength == 2 && size > (74+74*74)) || (maxLength ==
1 && size > 74))
        return BAD_LENGTH;
    string key;
    string value;
    unsigned int length;
    default_random_engine generator;
    uniform_int_distribution<int> lengthGen(1, maxLength);
    uniform_int_distribution<char> symbolsGen(48, 122);
    QStringList backup;
    int hFunc = (122+48)/2*maxLength; //midrange for worst1
    int keySum = 0;
    if (worstCase)
    {
        switch (hFuncNum)
        {
        case 1:
            for (int i=0; i<size; i++)
            {
                key.clear();
                value.clear();
                keySum = 0;
                while (keySum <= hFunc - 122*2)
                {
                    key += symbolsGen(generator);
                    keySum += key.back();
                }
                key += static_cast<char>((hFunc - keySum)/2);
                keySum += key.back();
```

```
            key += static_cast<char>(hFunc - keySum);
            length = static_cast<unsigned int>(lengthGen(generator));
            for (unsigned int j=0; j<length; j++)
            {
                value += symbolsGen(generator);
            }
            out << key << " " << value << endl;
        }
        break;
    case 2:
        length = static_cast<unsigned int>(maxLength);
        for (int i=0; i<size; i++)
        {
            key.clear();
            value.clear();
            key += 'a';
            for (unsigned int j=0; j<length-2; j++)
            {
                key += symbolsGen(generator);
                value += symbolsGen(generator);
            }
            key += 'z';
            out << key << " " << value << endl;
        }
        break;
    case 3:
        int i = 0;
        while (i<size)
        {
            length = 0;
            key.clear();
            value.clear();
            keySum = 0;
            while (keySum <= hFunc - 122*2)
            {
                key += symbolsGen(generator);
                keySum += key.back();
                length++;
            }
            key += static_cast<char>((hFunc - keySum)/2);
            keySum += key.back();
            key += static_cast<char>(hFunc - keySum);
            if (length+2 != static_cast<unsigned int>(maxLength))
                continue;
            length = static_cast<unsigned int>(lengthGen(generator));
            for (unsigned int j=0; j<length; j++)
            {
                value += symbolsGen(generator);
```

```
                }
                i++;
                out << key << " " << value << endl;
            }
        }
    }
    else
    {
        int i = 0;
        while (i<size)
        {
            key.clear();
            value.clear();
            length = static_cast<unsigned int>(lengthGen(generator));
            for (unsigned int j=0; j<length; j++)
            {
                key += symbolsGen(generator);
            }
            length = static_cast<unsigned int>(lengthGen(generator));
            for (unsigned int j=0; j<length; j++)
            {
                value += symbolsGen(generator);
            }
            if (backup.count(QString::fromStdString(key)))
                continue;
            backup.push_back(QString::fromStdString(key));
            i++;
            out << key << " " << value << endl;
        }
    }
    out.close();
    return 0;
}

QString StrStrWorker::getFromFile(QString fileName)
{
    ifstream fin(qPrintable(fileName), ios::in);
    string out;
    string temp;
    while (true)
    {
        getline(fin, temp);
        out += temp;
        if (!fin.eof())
            out += "\n";
        else
            break;
    }
```

```
    fin.close();
    return QString::fromStdString(out);
}

hCreateInfo StrStrWorker::createHashTable(QString input, int hFuncNum, int factor,
QString fileName)
{
    ofstream output(fileName.toStdString());
    if (!output)
    {
        info.error = true;
    }
    hTable.setSize(SIZE);
    hTable.clear();
    QStringList pairs = input.split("\n");
    if (pairs[pairs.length()-1].toStdString() == "")
        pairs.pop_back();
    QStringList temp;
    unsigned int iter = 0;
    unsigned int sumIter = 0;
    unsigned int maxIter = 0;
    unsigned int size = static_cast<unsigned int>(factor*pairs.length());
    for (int i=0; i<pairs.length(); i++)
    {
        temp = pairs[i].split(" ");
        if (temp.length() != 2)
        {
            if (!info.error)
            {
                output << "Error! Incorrect file content in line:\n\"";
                output << pairs[i].toStdString();
                output << "\"";
                info.error = true;
            }
            return info;
        }
        switch (hFuncNum)
        {
        case 1:
            iter   =   hTable.set(temp[0].toStdString(),   temp[1].toStdString(),
hFunc1(temp[0].toStdString(), size));
            break;
        case 2:
            iter   =   hTable.set(temp[0].toStdString(),   temp[1].toStdString(),
hFunc2(temp[0].toStdString(), size));
            break;
        case 3:
```

```
            iter   =   hTable.set(temp[0].toStdString(),   temp[1].toStdString(),
hFunc3(temp[0].toStdString(), size));
        }
        if (iter > maxIter)
            maxIter = iter;
        sumIter += iter;
    }
    info.hSize = hTable.getSize();
    info.itemCounter = hTable.getItemCounter();
    info.sumIterNum = sumIter;
    info.maxIterNum = maxIter;
    if (!info.error)
    {
        output << "Table length:\t";
        output << QString::number(info.hSize).toStdString();
        output << "\nNumber of items:\t";
        output << QString::number(info.itemCounter).toStdString();
        output << "\nAll inserts iterations:\t";
        output << QString::number(sumIter).toStdString();
        output << "\nMax insert iterations:\t";
        output << QString::number(maxIter).toStdString();
        output << "\n\n";
        output << QString::fromStdString(hTable.print()).toStdString();
        output.close();
    }
    info.error = false;
    return info;
}

unsigned int StrStrWorker::hFunc1(string key, unsigned int size)
{
    unsigned int result = 0;
    for (unsigned int i=0; i<key.length(); i++)
    {
        result += static_cast<unsigned char>(key[i]);
    }
    result %= size;
    return result;
}

unsigned int StrStrWorker::hFunc2(string key, unsigned int size)
{
    unsigned int result = 0;
    result += static_cast<unsigned char>(key.front());
    result += static_cast<unsigned char>(key.back());
    result *= key.length();
    result %= size;
    return result;
```

```
}

unsigned int StrStrWorker::hFunc3(string key, unsigned int size)
{
    unsigned int result = 0;
    if (key.length() > 0)
        result += static_cast<unsigned char>(key[0]);
    for (unsigned int i=1; i<key.length(); i++)
    {
        result += static_cast<unsigned char>(key[i]);
    }
    result *= key.length();
    result %= size;
    return result;
}


void StrStrWorker::saveStrToFile(QString output, QString fileName)
{
    ofstream fout(qPrintable(fileName));
    fout << qPrintable(output);
    fout.close();
}
```

ПРИЛОЖЕНИЕ Ж

ИСХОДНЫЙ КОД ПРОГРАММЫ. STRSTRTESTER.H

```
#ifndef STRSTRTESTER_H
#define STRSTRTESTER_H

#include "strstrworker.h"
#include "qcustomplot.h"

struct testInfo
{
    int min;
    int max;
    int step;
    int mode;
    QString fileName;
    bool checkMaxIter;
    bool worstCase;
    int maxLength;
    int size;
    int factor;
};


struct testResult
{
    unsigned int **maxIterArr;
    unsigned int **sumIterArr;
    long **clockArr;
    int **num;
};

int pairNumTester(StrStrWorker &ssw, testInfo info, QCustomPlot *&plot, string
&output, testResult &res);

#endif
```

ПРИЛОЖЕНИЕ И

ИСХОДНЫЙ КОД ПРОГРАММЫ. STRSTRTESTER.CPP

```
#include "strstrtester.h"

int pairNumTester(StrStrWorker &ssw, testInfo info, QCustomPlot *&plot, string
&output, testResult &res)
{
    if (info.max <= info.min)
        return BAD_MAXMIN;
    output.clear();
    int length = (info.max-info.min)/info.step;
    plot->graph(0)->data()->clear();
    plot->graph(1)->data()->clear();
    plot->graph(2)->data()->clear();
    if (info.checkMaxIter)
        plot->yAxis->setLabel("MI");
    else
        plot->yAxis->setLabel("I");
    if (info.worstCase)
    {
        plot->graph(0)->setPen(QPen(QBrush(QColor((Qt::red))), 8));
        plot->graph(1)->setPen(QPen(QBrush(QColor((Qt::green))), 5));
        plot->graph(2)->setPen(QPen(QBrush(QColor((Qt::blue))), 2));
    }
    else
    {
        plot->graph(0)->setPen(QPen(QBrush(QColor((Qt::red))), 3));
        plot->graph(1)->setPen(QPen(QBrush(QColor((Qt::green))), 3));
        plot->graph(2)->setPen(QPen(QBrush(QColor((Qt::blue))), 3));
    }
    res.maxIterArr = new unsigned int*[3];
    res.sumIterArr = new unsigned int*[3];
    res.clockArr = new long*[3];
    res.num = new int*[length+1];
    for (int i=0; i<3; i++)
    {
        res.maxIterArr[i] = new unsigned int[length+1];
        res.sumIterArr[i] = new unsigned int[length+1];
        res.clockArr[i] = new long[length+1];
        res.num[i] = new int[length+1];
    }
    hCreateInfo outInfo;
    int genErr = 0;
    int counter = -1;
    string param;
    long clStart, clEnd;
    switch (info.mode)
```

```
    {
    case 1:
        plot->axisRect()->insetLayout()->setInsetAlignment(0,
Qt::AlignRight|Qt::AlignTop);
        plot->xAxis->setLabel("L");
        for (int mL=info.min; mL<=info.max; mL+=info.step)
        {
            counter++;
            if (!info.worstCase)
                genErr   =   ssw.generateFile(info.size,   mL,   info.fileName,
info.worstCase, 0);
            for (int hF=1; hF<=3; hF++)
            {
                if (info.worstCase)
                    genErr  =  ssw.generateFile(info.size,  mL,  info.fileName,
info.worstCase, hF);
                if (genErr)
                    return genErr;
                clStart = clock();
                outInfo = ssw.createHashTable(ssw.getFromFile(info.fileName), hF,
info.factor, "");
                clEnd = clock() - clStart;
                res.maxIterArr[hF-1][counter] = outInfo.maxIterNum;
                res.sumIterArr[hF-1][counter] = outInfo.sumIterNum;
                res.num[hF-1][counter] = mL;
                res.clockArr[hF-1][counter] = clEnd;
                if (info.checkMaxIter)
                    plot->graph(hF-1)->addData(mL, outInfo.maxIterNum);
                else
                    plot->graph(hF-1)->addData(mL, outInfo.sumIterNum);
            }
            plot->rescaleAxes();
            plot->yAxis->setRangeLower(0);
            plot->replot();
        }
        param = "Key max length";
        break;
    case 2:
        plot->xAxis->setLabel("N");
        for (int sz=info.min; sz<=info.max; sz+=info.step)
        {
            counter++;
            if (!info.worstCase)
                genErr  =  ssw.generateFile(sz,  info.maxLength,  info.fileName,
info.worstCase, 0);
            for (int hF=1; hF<=3; hF++)
            {
                if (info.worstCase)
```

```
                    genErr = ssw.generateFile(sz, info.maxLength, info.fileName,
info.worstCase, hF);
                if (genErr)
                    return genErr;
                clStart = clock();
                outInfo = ssw.createHashTable(ssw.getFromFile(info.fileName), hF,
info.factor, "");
                clEnd = clock() - clStart;
                res.maxIterArr[hF-1][counter] = outInfo.maxIterNum;
                res.sumIterArr[hF-1][counter] = outInfo.sumIterNum;
                res.num[hF-1][counter] = sz;
                res.clockArr[hF-1][counter] = clEnd;
                if (info.checkMaxIter)
                    plot->graph(hF-1)->addData(sz, outInfo.maxIterNum);
                else
                    plot->graph(hF-1)->addData(sz, outInfo.sumIterNum);
            }
            plot->rescaleAxes();
            plot->yAxis->setRangeLower(0);
            plot->replot();
        }
        param = "Pair number";
        break;
    case 3:
        plot->axisRect()->insetLayout()->setInsetAlignment(0,
Qt::AlignRight|Qt::AlignBottom);
        plot->xAxis->setLabel("X");
        for (int fc=info.min; fc<=info.max; fc+=info.step)
        {
            counter++;
            if (!info.worstCase)
                genErr     =     ssw.generateFile(info.size,     info.maxLength,
info.fileName, info.worstCase, 0);
            for (int hF=1; hF<=3; hF++)
            {
                if (info.worstCase)
                    genErr    =    ssw.generateFile(info.size,    info.maxLength,
info.fileName, info.worstCase, hF);
                if (genErr)
                    return genErr;
                clStart = clock();
                outInfo = ssw.createHashTable(ssw.getFromFile(info.fileName), hF,
fc, "");
                clEnd = clock() - clStart;
                res.maxIterArr[hF-1][counter] = outInfo.maxIterNum;
                res.sumIterArr[hF-1][counter] = outInfo.sumIterNum;
                res.num[hF-1][counter] = fc;
                res.clockArr[hF-1][counter] = clEnd;
```

```
                if (info.checkMaxIter)
                    plot->graph(hF-1)->addData(fc, outInfo.maxIterNum);
                else
                    plot->graph(hF-1)->addData(fc, outInfo.sumIterNum);
            }
            plot->rescaleAxes();
            plot->yAxis->setRangeLower(0);
            plot->replot();
        }
        param = "Size factor  ";
    }
    output += "Result for hFunc1:\n";
    output += param + "\tWorst hF1\tAll hF1\tTimehF1\tWorst hF2\tAll hF2\tTime
hF2\tWorst hF3\tAll hF3\tTime hF3\n";
    for (int i=0; i<=counter; i++)
    {
        output      +=      to_string(res.num[0][i])      +      "\t\t"      +
to_string(res.maxIterArr[0][i]) + "\t" + to_string(res.sumIterArr[0][i]) + "\t"
+ to_string(res.clockArr[0][i]);
        output   +=   "\t"   +   to_string(res.maxIterArr[1][i])   +   "\t"   +
to_string(res.sumIterArr[1][i]) + "\t" + to_string(res.clockArr[1][i]);
        output   +=   "\t"   +   to_string(res.maxIterArr[2][i])   +   "\t"   +
to_string(res.sumIterArr[2][i]) + "\t" + to_string(res.clockArr[2][i]) + "\n";
    }
    return 0;
}
```

ПРИЛОЖЕНИЕ К

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAINWINDOW.H

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
#include "strstrtester.h"

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:

    void on_GPushOk_clicked();

    void on_HPushOk_clicked();

    void on_MPushOk_clicked();

    void on_OutSetPlot_toggled(bool checked);

    void on_OutSaveFile_clicked();

private:
    void setupPlot();
    Ui::MainWindow *ui;
    StrStrWorker ssw;
};
#endif // MAINWINDOW_H
```

ПРИЛОЖЕНИЕ Л

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAINWINDOW.CPP

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "inout.h"

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->outputText->setReadOnly(true);
    ui->OutSetPlot->setChecked(true);
    ui->MTestSum->setChecked(true);
    ui->MTestPairs->setChecked(true);
    setupPlot();
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::setupPlot()
{
    ui->outputPlot->addGraph();
    ui->outputPlot->addGraph();
    ui->outputPlot->addGraph();
    ui->outputPlot->graph(0)->setName("hFunc1");
    ui->outputPlot->graph(1)->setName("hFunc2");
    ui->outputPlot->graph(2)->setName("hFunc3");
    ui->outputPlot->graph(0)->setPen(QPen(QBrush(QColor((Qt::red))), 3));
    ui->outputPlot->graph(1)->setPen(QPen(QBrush(QColor((Qt::green))), 3));
    ui->outputPlot->graph(2)->setPen(QPen(QBrush(QColor((Qt::blue))), 3));
    ui->outputPlot->setInteractions(QCP::iRangeZoom    |    QCP::iRangeDrag    |
QCP::iSelectLegend);
    ui->outputPlot->axisRect()->setRangeZoom(Qt::Vertical);
    ui->outputPlot->axisRect()->setRangeDrag(Qt::Vertical);
    ui->outputPlot->axisRect()->insetLayout()->setInsetAlignment(0,
Qt::AlignLeft|Qt::AlignTop);
    ui->outputPlot->legend->setVisible(true);
    QSharedPointer<QCPAxisTicker> ticker(new QCPAxisTicker);
    ticker->setTickCount(20);
    ui->outputPlot->xAxis->setTicker(ticker);
    ui->outputPlot->yAxis->setTicker(ticker);
}
```

```
void MainWindow::on_GPushOk_clicked()
{
    int pairs = ui->GInputPairs->toPlainText().toInt();
    int length = ui->GInputLength->toPlainText().toInt();
    QString fileName = ui->GInputFile->toPlainText();
    QMessageBox message;
    int      res      =      ssw.generateFile(pairs,      length,      fileName,
ui->GCheckWorst->isChecked(), ui->spinHFunc->value());
    switch (res)
    {
    case BAD_SIZE:
        message.setText("Error! Pairs number input is not correct.");
        message.exec();
        break;
    case BAD_LENGTH:
        message.setText("Error! Words length input is not correct.");
        message.exec();
        break;
    case BAD_FILE:
        message.setText("Error! File name input is not correct.");
        message.exec();
        break;
    default:
        message.setText("File was created.");
        message.exec();
    }
    ui->HInputFile->setText(ui->GInputFile->toPlainText());
}

void MainWindow::on_HPushOk_clicked()
{
    QString fileName = ui->HInputFile->toPlainText();
    QString output = ui->HInputOut->toPlainText();
    int factor = ui->spinFactor->value();
    QMessageBox errOut;
    hCreateInfo    info    =    ssw.createHashTable(ssw.getFromFile(fileName),
ui->spinHFunc->value(), factor, output);
    if (info.error)
    {
        errOut.setText("Error!");
        errOut.exec();
    }
    else
        ui->outputText->setText(ssw.getFromFile(output));
}

void MainWindow::on_MPushOk_clicked()
{
```

```
    testInfo info;
    testResult res;
    QMessageBox message;
    string output;
    if                (ui->MInputMin->toPlainText().isEmpty()                ||
ui->MInputMax->toPlainText().isEmpty())
    {
        message.setText("Error! Min or max input is not correct");
        message.exec();
        return;
    }
    ui->OutSetPlot->setChecked(true);
    info.min = ui->MInputMin->toPlainText().toInt();
    info.max = ui->MInputMax->toPlainText().toInt();
    info.step = ui->spinStep->value();
    info.fileName = "test.txt";
    info.checkMaxIter = ui->MTestMax->isChecked();
    info.worstCase = ui->GCheckWorst->isChecked();
    if (ui->MTestKey->isChecked())
    {
        info.mode = 1;
        info.maxLength = 0;
        info.size = ui->GInputPairs->toPlainText().toInt();
        info.factor = ui->spinFactor->value();
    }
    else if (ui->MTestPairs->isChecked())
    {
        info.mode = 2;
        info.size = 0;
        info.maxLength = ui->GInputLength->toPlainText().toInt();
        info.factor = ui->spinFactor->value();
    }
    else
    {
        info.mode = 3;
        info.factor = 0;
        info.size = ui->GInputPairs->toPlainText().toInt();
        info.maxLength = ui->GInputLength->toPlainText().toInt();
    }
    int err = pairNumTester(ssw, info, ui->outputPlot, output, res);
    switch (err)
    {
    case BAD_SIZE:
        message.setText("Error! Pairs number input is not correct.");
        message.exec();
        break;
    case BAD_LENGTH:
        message.setText("Error! Words length input is not correct.");
```

```
        message.exec();
        break;
    case BAD_FILE:
        message.setText("Error! File name input is not correct.");
        message.exec();
        break;
    case BAD_MAXMIN:
        message.setText("Error! Max should be greater than min.");
        message.exec();
        break;
    default:
        message.setText("Plot was created.");
        message.exec();
    }
    ui->outputText->setText(QString::fromStdString(output));
    ui->outputPlot->replot();
}

void MainWindow::on_OutSetPlot_toggled(bool checked)
{
    if (checked)
        ui->outputText->hide();
    else
        ui->outputText->show();
}

void MainWindow::on_OutSaveFile_clicked()
{
    QString filePath = QFileDialog::getSaveFileName(this, tr("Save to TXT file"),
QDir::homePath(), tr("*.txt"));
    if (QString::compare(filePath, QString()) != 0)
    {
        QMessageBox message;
        ssw.saveStrToFile(ui->outputText->toPlainText(), filePath);
        message.setText("Saved to file");
        message.exec();
    }
}
```

ПРИЛОЖЕНИЕ М

ИСХОДНЫЙ КОД ПРОГРАММЫ. LIBRARIES.H

```
#ifndef LIBRARIES_H
#define LIBRARIES_H

#define SIZE_LIMIT 50000

#define BAD_FILE 1
#define BAD_SIZE 2
#define BAD_LENGTH 3
#define BAD_MAXMIN 4

#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <QMainWindow>
#include <QGraphicsItem>
#include <QGraphicsView>
#include <QGraphicsEffect>
#include <QFileDialog>
#include <QStandardPaths>
#include <QtGui>
#include <QLabel>
#include <QColorDialog>
#include <QInputDialog>
#include <QMainWindow>
#include <QPushButton>
#include <QMessageBox>
#include <QStringList>
#include <QTextEdit>
#include <cstdlib>
#include <random>
#include <ctime>
using namespace std;

#endif // LIBRARIES_H
```

ПРИЛОЖЕНИЕ Н

ИСХОДНЫЙ КОД ПРОГРАММЫ. LR5.PRO

```
QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets printsupport

CONFIG += c++11

# The following define makes your compiler emit warnings if you use
# any Qt feature that has been marked deprecated (the exact warnings
# depend on your compiler). Please consult the documentation of the
# deprecated API in order to know how to port your code away from it.
DEFINES += QT_DEPRECATED_WARNINGS

# You can also make your code fail to compile if it uses deprecated APIs.
# In order to do so, uncomment the following line.
# You can also select to disable deprecated APIs only up to a certain version of
Qt.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000     # disables all the APIs
deprecated before Qt 6.0.0
SOURCES += \
    main.cpp \
    mainwindow.cpp \
    qcustomplot.cpp \
    strstrtester.cpp \
    strstrworker.cpp

HEADERS += \
    cvector.h \
    hashtable.h \
    libraries.h \
    mainwindow.h \
    pair.h \
    qcustomplot.h \
    strstrtester.h \
    strstrworker.h

FORMS += \
    mainwindow.ui

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target

RESOURCES += \
    images/images.qrc
```

ПРИЛОЖЕНИЕ О

ИСХОДНЫЙ КОД ПРОГРАММЫ. MAINWINDOW.UI

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>1280</width>
    <height>1000</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <property name="styleSheet">
   <string notr="true">background-color: rgb(30, 30, 30)</string>
  </property>
  <widget class="QWidget" name="centralwidget">
   <widget class="QPushButton" name="GPushOk">
    <property name="geometry">
     <rect>
      <x>270</x>
      <y>100</y>
      <width>50</width>
      <height>50</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">border-image: url(:/ok.png);</string>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
   <widget class="QTextEdit" name="GInputPairs">
    <property name="geometry">
     <rect>
      <x>120</x>
      <y>50</y>
      <width>131</width>
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
```

```
     <string notr="true">background-color: rgb(50, 50, 50);
color: white;
border-radius: 15px;
font: 10pt &quot;Tahoma&quot;;
padding: 4px 5px 0px 7px;
border: 2px solid rgb(80,80,80);</string>
    </property>
    <property name="html">
     <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot;
&quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot;1&quot;
/&gt;&lt;style type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
&lt;/style&gt;&lt;/head&gt;&lt;body  style=&quot;  font-family:'Tahoma';  font-
size:10pt; font-weight:400; font-style:normal;&quot;&gt;
&lt;p  style=&quot;-qt-paragraph-type:empty;  margin-top:0px;  margin-bottom:0px;
margin-left:0px;       margin-right:0px;       -qt-block-indent:0;       text-
indent:0px;&quot;&gt;&lt;br /&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
    </property>
   </widget>
   <widget class="QLabel" name="GNameMain">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>10</y>
      <width>331</width>
      <height>31</height>
     </rect>
    </property>
    <property name="font">
     <font>
      <family>Tahoma</family>
      <pointsize>11</pointsize>
      <weight>75</weight>
      <italic>false</italic>
      <bold>true</bold>
     </font>
    </property>
    <property name="layoutDirection">
     <enum>Qt::LeftToRight</enum>
    </property>
    <property name="styleSheet">
     <string notr="true">font: bold 11pt &quot;Tahoma&quot;;
color: rgb(235, 235, 235);</string>
    </property>
    <property name="text">
     <string>Generate file</string>
    </property>
```

```
    <property name="alignment">
     <set>Qt::AlignCenter</set>
    </property>
   </widget>
   <widget class="QLabel" name="HNameMain">
    <property name="geometry">
     <rect>
      <x>360</x>
      <y>10</y>
      <width>611</width>
      <height>31</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: bold 11pt &quot;Tahoma&quot;;
color: rgb(235, 235, 235);</string>
    </property>
    <property name="text">
     <string>Hash table file</string>
    </property>
    <property name="alignment">
     <set>Qt::AlignCenter</set>
    </property>
   </widget>
   <widget class="QPushButton" name="HPushOk">
    <property name="geometry">
     <rect>
      <x>920</x>
      <y>70</y>
      <width>50</width>
      <height>50</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">border-image: url(:/ok.png);</string>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
   <widget class="QTextEdit" name="GInputLength">
    <property name="geometry">
     <rect>
      <x>120</x>
      <y>105</y>
      <width>131</width>
      <height>41</height>
     </rect>
```

```
    </property>
    <property name="styleSheet">
     <string notr="true">background-color: rgb(50, 50, 50);
color: white;
border-radius: 15px;
font: 10pt &quot;Tahoma&quot;;
padding: 4px 5px 0px 7px;
border: 2px solid rgb(80,80,80);</string>
    </property>
    <property name="html">
     <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot;
&quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot;1&quot;
/&gt;&lt;style type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-family:'Tahoma'; font-
size:10pt; font-weight:400; font-style:normal;&quot;&gt;
&lt;p style=&quot;-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px;
margin-left:0px; margin-right:0px; -qt-block-indent:0; text-
indent:0px;&quot;&gt;&lt;br /&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
    </property>
   </widget>
   <widget class="QLabel" name="GNamePairs">
    <property name="geometry">
     <rect>
      <x>10</x>
      <y>50</y>
      <width>101</width>
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 10pt &quot;Tahoma&quot;;
color: rgb(235, 235, 235);</string>
    </property>
    <property name="text">
     <string>Pairs num:</string>
    </property>
    <property name="alignment">
     <set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>
    </property>
   </widget>
   <widget class="QLabel" name="GNameLength">
    <property name="geometry">
     <rect>
      <x>10</x>
      <y>105</y>
      <width>101</width>
```

```
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 10pt &amp;quot;Tahoma&amp;quot;;
color: rgb(235, 235, 235);</string>
    </property>
    <property name="text">
     <string>Key length:</string>
    </property>
    <property name="alignment">
     <set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>
    </property>
   </widget>
   <widget class="QLabel" name="GNameFile">
    <property name="geometry">
     <rect>
      <x>10</x>
      <y>160</y>
      <width>101</width>
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 10pt &amp;quot;Tahoma&amp;quot;;
color: rgb(235, 235, 235);</string>
    </property>
    <property name="text">
     <string>File name:</string>
    </property>
    <property name="alignment">
     <set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>
    </property>
   </widget>
   <widget class="QTextEdit" name="GInputFile">
    <property name="geometry">
     <rect>
      <x>120</x>
      <y>160</y>
      <width>131</width>
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">background-color: rgb(50, 50, 50);
color: white;
border-radius: 15px;
font: 10pt &amp;quot;Tahoma&amp;quot;;
```

```
padding: 4px 5px 0px 7px;
border: 2px solid rgb(80,80,80);</string>
    </property>
    <property name="html">
     <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot;
&quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot;1&quot;
/&gt;&lt;style type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-family:'Tahoma'; font-
size:10pt; font-weight:400; font-style:normal;&quot;&gt;
&lt;p style=&quot;-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px;
margin-left:0px; margin-right:0px; -qt-block-indent:0; text-
indent:0px;&quot;&gt;&lt;br /&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
    </property>
   </widget>
   <widget class="QLabel" name="HNameFile">
    <property name="geometry">
     <rect>
      <x>600</x>
      <y>50</y>
      <width>101</width>
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 10pt &quot;Tahoma&quot;;
color: rgb(235, 235, 235);</string>
    </property>
    <property name="text">
     <string>Input file:</string>
    </property>
    <property name="alignment">
     <set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>
    </property>
   </widget>
   <widget class="Line" name="VLine">
    <property name="geometry">
     <rect>
      <x>340</x>
      <y>25</y>
      <width>3</width>
      <height>310</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">background-color: rgb(70, 70, 70);</string>
    </property>
```

```
    <property name="orientation">
     <enum>Qt::Vertical</enum>
    </property>
   </widget>
   <widget class="Line" name="HLine">
    <property name="geometry">
     <rect>
      <x>365</x>
      <y>160</y>
      <width>600</width>
      <height>3</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">background-color: rgb(70, 70, 70);</string>
    </property>
    <property name="orientation">
     <enum>Qt::Horizontal</enum>
    </property>
   </widget>
   <widget class="QLabel" name="MNameMain">
    <property name="geometry">
     <rect>
      <x>360</x>
      <y>180</y>
      <width>611</width>
      <height>31</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: bold 11pt &quot;Tahoma&quot;;
color: rgb(235, 235, 235);</string>
    </property>
    <property name="text">
     <string>Graphmaker</string>
    </property>
    <property name="alignment">
     <set>Qt::AlignCenter</set>
    </property>
   </widget>
   <widget class="QCheckBox" name="GCheckWorst">
    <property name="geometry">
     <rect>
      <x>60</x>
      <y>230</y>
      <width>191</width>
      <height>22</height>
     </rect>
```

```
    </property>
    <property name="layoutDirection">
     <enum>Qt::LeftToRight</enum>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 10pt &quot;Tahoma&quot;;
color: rgb(235, 235, 235);</string>
    </property>
    <property name="text">
     <string>Worst case for HFunc</string>
    </property>
   </widget>
   <widget class="QLabel" name="MNameMin">
    <property name="geometry">
     <rect>
      <x>370</x>
      <y>255</y>
      <width>41</width>
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 10pt &quot;Tahoma&quot;;
color: rgb(235, 235, 235);</string>
    </property>
    <property name="text">
     <string>Min:</string>
    </property>
    <property name="alignment">
     <set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>
    </property>
   </widget>
   <widget class="QTextEdit" name="MInputMin">
    <property name="geometry">
     <rect>
      <x>420</x>
      <y>255</y>
      <width>131</width>
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">background-color: rgb(50, 50, 50);
color: white;
border-radius: 15px;
font: 10pt &quot;Tahoma&quot;;
padding: 4px 5px 0px 7px;
border: 2px solid rgb(80,80,80);</string>
```

```
    </property>
    <property name="html">
     <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot;
&quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot;1&quot;
/&gt;&lt;style type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-family:'Tahoma'; font-
size:10pt; font-weight:400; font-style:normal;&quot;&gt;
&lt;p style=&quot;-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px;
margin-left:0px; margin-right:0px; -qt-block-indent:0; text-
indent:0px;&quot;&gt;&lt;br /&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
    </property>
   </widget>
   <widget class="QLabel" name="MNameMax">
    <property name="geometry">
     <rect>
      <x>570</x>
      <y>255</y>
      <width>41</width>
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 10pt &quot;Tahoma&quot;;
color: rgb(235, 235, 235);</string>
    </property>
    <property name="text">
     <string>Max:</string>
    </property>
    <property name="alignment">
     <set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>
    </property>
   </widget>
   <widget class="QTextEdit" name="MInputMax">
    <property name="geometry">
     <rect>
      <x>620</x>
      <y>255</y>
      <width>131</width>
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">background-color: rgb(50, 50, 50);
color: white;
border-radius: 15px;
font: 10pt &quot;Tahoma&quot;;
```

```
padding: 4px 5px 0px 7px;
border: 2px solid rgb(80,80,80);</string>
    </property>
    <property name="html">
     <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot;
&quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot;1&quot;
/&gt;&lt;style type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-family:'Tahoma'; font-
size:10pt; font-weight:400; font-style:normal;&quot;&gt;
&lt;p style=&quot;-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px;
margin-left:0px; margin-right:0px; -qt-block-indent:0; text-
indent:0px;&quot;&gt;&lt;br /&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
    </property>
   </widget>
   <widget class="QLabel" name="HNameHFunc">
    <property name="geometry">
     <rect>
      <x>370</x>
      <y>50</y>
      <width>131</width>
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 10pt &quot;Tahoma&quot;;
color: rgb(235, 235, 235);</string>
    </property>
    <property name="text">
     <string>HFunc number:</string>
    </property>
    <property name="alignment">
     <set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>
    </property>
   </widget>
   <widget class="QTextEdit" name="HInputFile">
    <property name="geometry">
     <rect>
      <x>710</x>
      <y>50</y>
      <width>131</width>
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">background-color: rgb(50, 50, 50);
color: white;
```

```
border-radius: 15px;
font: 10pt &quot;Tahoma&quot;;
padding: 4px 5px 0px 7px;
border: 2px solid rgb(80,80,80);</string>
    </property>
    <property name="html">
     <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot;
&quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot;1&quot;
/&gt;&lt;style type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-family:'Tahoma'; font-
size:10pt; font-weight:400; font-style:normal;&quot;&gt;
&lt;p style=&quot;-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px;
margin-left:0px; margin-right:0px; -qt-block-indent:0; text-
indent:0px;&quot;&gt;&lt;br /&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
    </property>
   </widget>
   <widget class="QPushButton" name="MPushOk">
    <property name="geometry">
     <rect>
      <x>920</x>
      <y>290</y>
      <width>50</width>
      <height>50</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">border-image: url(:/ok.png);</string>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
   <widget class="QSpinBox" name="spinHFunc">
    <property name="geometry">
     <rect>
      <x>510</x>
      <y>50</y>
      <width>43</width>
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 11pt &quot;Tahoma&quot;;
color: rgb(235, 235, 235);</string>
    </property>
    <property name="minimum">
```

```
      <number>1</number>
     </property>
     <property name="maximum">
      <number>3</number>
     </property>
    </widget>
    <widget class="QCustomPlot" name="outputPlot" native="true">
     <property name="geometry">
      <rect>
       <x>30</x>
       <y>440</y>
       <width>1220</width>
       <height>500</height>
      </rect>
     </property>
     <property name="styleSheet">
      <string notr="true"/>
     </property>
     <widget class="QTextEdit" name="outputText">
      <property name="geometry">
       <rect>
        <x>0</x>
        <y>0</y>
        <width>1220</width>
        <height>500</height>
       </rect>
      </property>
      <property name="styleSheet">
       <string notr="true">background-color: rgb(40, 40, 40);
color: white;
font: 10pt &amp;quot;Tahoma&amp;quot;;
padding: 4px 5px 0px 7px;</string>
      </property>
     </widget>
    </widget>
    <widget class="QTextEdit" name="HInputOut">
     <property name="geometry">
      <rect>
       <x>710</x>
       <y>105</y>
       <width>131</width>
       <height>41</height>
      </rect>
     </property>
     <property name="styleSheet">
      <string notr="true">background-color: rgb(50, 50, 50);
color: white;
border-radius: 15px;
```

```
font: 10pt &quot;Tahoma&quot;;
padding: 4px 5px 0px 7px;
border: 2px solid rgb(80,80,80);</string>
    </property>
    <property name="html">
     <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot;
&quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot;1&quot;
/&gt;&lt;style type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
&lt;/style&gt;&lt;/head&gt;&lt;body  style=&quot;  font-family:'Tahoma';  font-
size:10pt; font-weight:400; font-style:normal;&quot;&gt;
&lt;p  style=&quot;-qt-paragraph-type:empty;  margin-top:0px;  margin-bottom:0px;
margin-left:0px;       margin-right:0px;       -qt-block-indent:0;       text-
indent:0px;&quot;&gt;&lt;br /&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
    </property>
   </widget>
   <widget class="QLabel" name="HNameOut">
    <property name="geometry">
     <rect>
      <x>600</x>
      <y>105</y>
      <width>101</width>
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 10pt &quot;Tahoma&quot;;
color: rgb(235, 235, 235);</string>
    </property>
    <property name="text">
     <string>Output file:</string>
    </property>
    <property name="alignment">
     <set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>
    </property>
   </widget>
   <widget class="QLabel" name="HNameLength">
    <property name="geometry">
     <rect>
      <x>370</x>
      <y>105</y>
      <width>71</width>
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 10pt &quot;Tahoma&quot;;
```

```
color: rgb(235, 235, 235);</string>
    </property>
    <property name="text">
     <string>Factor:</string>
    </property>
    <property name="alignment">
     <set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>
    </property>
   </widget>
   <widget class="QSpinBox" name="spinFactor">
    <property name="geometry">
     <rect>
      <x>452</x>
      <y>105</y>
      <width>101</width>
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 11pt &quot;Tahoma&quot;;
color: rgb(235, 235, 235);</string>
    </property>
    <property name="minimum">
     <number>1</number>
    </property>
    <property name="maximum">
     <number>1000</number>
    </property>
    <property name="value">
     <number>1</number>
    </property>
   </widget>
   <widget class="QRadioButton" name="MTestPairs">
    <property name="geometry">
     <rect>
      <x>440</x>
      <y>220</y>
      <width>141</width>
      <height>22</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 10pt &quot;Tahoma&quot;;
color: rgb(235, 235, 235);</string>
    </property>
    <property name="text">
     <string>Pairs num</string>
    </property>
```

```
   </widget>
   <widget class="QRadioButton" name="MTestKey">
    <property name="geometry">
     <rect>
      <x>600</x>
      <y>220</y>
      <width>131</width>
      <height>22</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 10pt &quot;Tahoma&quot;;
color: rgb(235, 235, 235);</string>
    </property>
    <property name="text">
     <string>Key length</string>
    </property>
   </widget>
   <widget class="QRadioButton" name="MTestFactor">
    <property name="geometry">
     <rect>
      <x>760</x>
      <y>220</y>
      <width>131</width>
      <height>22</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 10pt &quot;Tahoma&quot;;
color: rgb(235, 235, 235);</string>
    </property>
    <property name="text">
     <string>Factor</string>
    </property>
   </widget>
   <widget class="QSpinBox" name="spinStep">
    <property name="geometry">
     <rect>
      <x>842</x>
      <y>255</y>
      <width>61</width>
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 11pt &quot;Tahoma&quot;;
color: rgb(235, 235, 235);</string>
    </property>
```

```
    <property name="minimum">
     <number>1</number>
    </property>
    <property name="maximum">
     <number>1000</number>
    </property>
    <property name="value">
     <number>1</number>
    </property>
   </widget>
   <widget class="QLabel" name="HNameStep">
    <property name="geometry">
     <rect>
      <x>770</x>
      <y>255</y>
      <width>51</width>
      <height>41</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 10pt &quot;Tahoma&quot;;
color: rgb(235, 235, 235);</string>
    </property>
    <property name="text">
     <string>Step:</string>
    </property>
    <property name="alignment">
     <set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>
    </property>
   </widget>
   <widget class="QWidget" name="horizontalLayoutWidget">
    <property name="geometry">
     <rect>
      <x>480</x>
      <y>310</y>
      <width>361</width>
      <height>31</height>
     </rect>
    </property>
    <layout class="QHBoxLayout" name="MLayout">
     <item>
      <widget class="QRadioButton" name="MTestSum">
       <property name="styleSheet">
        <string notr="true">font: 10pt &quot;Tahoma&quot;;
color: rgb(235, 235, 235);</string>
       </property>
       <property name="text">
        <string>Iterations sum</string>
```

```
       </property>
      </widget>
     </item>
     <item>
      <widget class="QRadioButton" name="MTestMax">
       <property name="layoutDirection">
        <enum>Qt::RightToLeft</enum>
       </property>
       <property name="styleSheet">
        <string notr="true">font: 10pt &amp;quot;Tahoma&amp;quot;;
color: rgb(235, 235, 235);</string>
       </property>
       <property name="text">
        <string>Longest insert</string>
       </property>
      </widget>
     </item>
    </layout>
   </widget>
   <widget class="Line" name="OutLine">
    <property name="geometry">
     <rect>
      <x>30</x>
      <y>360</y>
      <width>1220</width>
      <height>3</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">background-color: rgb(70, 70, 70);</string>
    </property>
    <property name="orientation">
     <enum>Qt::Horizontal</enum>
    </property>
   </widget>
   <widget class="QWidget" name="horizontalLayoutWidget_2">
    <property name="geometry">
     <rect>
      <x>30</x>
      <y>390</y>
      <width>281</width>
      <height>31</height>
     </rect>
    </property>
    <layout class="QHBoxLayout" name="OutLayout">
     <item>
      <widget class="QRadioButton" name="OutSetPlot">
       <property name="styleSheet">
```

```
        <string notr="true">font: 10pt &quot;Tahoma&quot;;
color: rgb(235, 235, 235);</string>
       </property>
       <property name="text">
        <string>Plot output</string>
       </property>
      </widget>
     </item>
     <item>
      <widget class="QRadioButton" name="OutSetText">
       <property name="layoutDirection">
        <enum>Qt::RightToLeft</enum>
       </property>
       <property name="styleSheet">
        <string notr="true">font: 10pt &quot;Tahoma&quot;;
color: rgb(235, 235, 235);</string>
       </property>
       <property name="text">
        <string>Text output</string>
       </property>
      </widget>
     </item>
    </layout>
   </widget>
   <widget class="QPushButton" name="OutSaveFile">
    <property name="geometry">
     <rect>
      <x>1200</x>
      <y>380</y>
      <width>50</width>
      <height>50</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">border-image: url(:/savef.png);</string>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
   <zorder>GPushOk</zorder>
   <zorder>GInputPairs</zorder>
   <zorder>GNameMain</zorder>
   <zorder>HNameMain</zorder>
   <zorder>HPushOk</zorder>
   <zorder>GInputLength</zorder>
   <zorder>GNamePairs</zorder>
   <zorder>GNameLength</zorder>
```

```
  <zorder>GNameFile</zorder>
  <zorder>GInputFile</zorder>
  <zorder>HNameFile</zorder>
  <zorder>VLine</zorder>
  <zorder>HLine</zorder>
  <zorder>MNameMain</zorder>
  <zorder>GCheckWorst</zorder>
  <zorder>MNameMin</zorder>
  <zorder>MInputMin</zorder>
  <zorder>MNameMax</zorder>
  <zorder>MInputMax</zorder>
  <zorder>HNameHFunc</zorder>
  <zorder>HInputFile</zorder>
  <zorder>MPushOk</zorder>
  <zorder>spinHFunc</zorder>
  <zorder>HInputOut</zorder>
  <zorder>HNameOut</zorder>
  <zorder>HNameLength</zorder>
  <zorder>spinFactor</zorder>
  <zorder>MTestPairs</zorder>
  <zorder>MTestKey</zorder>
  <zorder>MTestFactor</zorder>
  <zorder>spinStep</zorder>
  <zorder>HNameStep</zorder>
  <zorder>outputPlot</zorder>
  <zorder>horizontalLayoutWidget</zorder>
  <zorder>OutLine</zorder>
  <zorder>horizontalLayoutWidget_2</zorder>
  <zorder>OutSaveFile</zorder>
 </widget>
 <widget class="QMenuBar" name="menubar">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>1280</width>
    <height>25</height>
   </rect>
  </property>
 </widget>
 <widget class="QStatusBar" name="statusbar"/>
</widget>
<customwidgets>
 <customwidget>
  <class>QCustomPlot</class>
  <extends>QWidget</extends>
  <header>qcustomplot.h</header>
  <container>1</container>
```

```
  </customwidget>
 </customwidgets>
 <resources/>
 <connections/>
</ui>
```