

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра информационной безопасности

КУРСОВАЯ РАБОТА
по дисциплине «Алгоритмы и структуры данных»
Тема: Алгоритмы повышенной сложности

Студентка гр. 1361

Токарева У.В.

Преподаватель

Беляев А.В.

Санкт-Петербург

2022

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студентка Токарева У. В.

Группа 1361

Тема работы: Поиск одинаковых файлов в файловой системе на основе подсчета хеш-сумм

Исходные данные:

Программе указывается каталог, для всех файлов которого (включая размещенные в его подкаталогах) программа должна вычислить любой из современных некриптостойких хеш-функций разрядности не менее 64 бит. В выводе на экран программа должна сгруппировать файлы по вычисленному хешу и вывести группы файлов, содержащие не менее 2 записей (уникальные файлы отображать не требуется), показывая путь от стартового каталога, размер файла, значение хеша в 16-ричном виде.

Содержание пояснительной записки:

Введение, хеш, зачем нужны хеши файлов, некриптографические хеш-функции, реализация программы, основные сведения о программном обеспечении, реализованные функции, результат тестирования программы, заключение, список использованных источников, приложение 1. Руководство пользователя, приложение 2. Исходный код программы.

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 05.11.2022

Дата сдачи реферата: 28.12.2022

Студентка гр. 1361

Токарева У.В.

Преподаватель

Беляев А.В.

АННОТАЦИЯ

Основная задача проекта – разработка программы, предназначенной для поиска одинаковых файлов в директории. Мной была выбрана некриптографическая хеш-функция djb2. Благодаря использованию данной функции мне удалось в значительной степени увеличить скорость работы программы.

SUMMARY

The main objective of the project is to develop a program designed to search for identical files in a directory. I chose the non-cryptographic hash function djb2. Thanks to the use of this function, I was able to significantly increase the speed of the program.

СОДЕРЖАНИЕ

Введение	6
1. Хеш	7
1.1. Зачем нужны хеши файлов	7
1.2. Некриптографические хеш-функции	7
2. Реализация программы	8
2.1. Основные сведения о программном обеспечении	8
2.2. Реализованные функции	8
3. Результат тестирования программы	9
Заключение	10
Список использованных источников	11
Приложение 1. Руководство пользователя	12
Приложение 2. Исходный код программы	13

ВВЕДЕНИЕ

Основной задачей работы является разработка программного кода, который обеспечивает поиск одинаковых файлов в папке. Программа способна обрабатывать файлы, находящиеся не только в самой директории, но и во вложенных директориях.

Для выполнения работы мне потребовалось найти необходимую информацию в литературе и интернет-ресурсах и, после написания программы, проверить ее работу с помощью одной из сред программирования.

1. ХЕШ

1.1. Зачем нужны хеши файлов

Хеш-функцией называется математическое преобразование информации в короткую, определенной длины строку.

Анализ при помощи хеш-функций часто используют для контроля целостности важных файлов операционной системы, важных программ, важных данных. Контроль может производиться как по необходимости, так и на регулярной основе.

Вначале определяют, целостность каких файлов нужно контролировать. Для каждого файла производится вычисления значения его хеша по специальному алгоритму с сохранением результата. Через некоторое время производится аналогичный расчет и сравниваются результаты. Если значения отличаются, значит информация содержащаяся в файле была изменена.

1.2. Некриптографические хеш-функции

Если криптографические хеш-функции у всех на слуху, то про некриптографические (хеш-функции общего назначения) известно мало. Некриптографические функции применяются там, где на данные не воздействуют третьи лица (злоумышленник). Например, такие функции могут использоваться для построения хеш-таблиц.

2. РЕАЛИЗАЦИЯ ПРОГРАММЫ

2.1. Основные сведения о программном обеспечении

Для написания курсовой работы мною был выбран язык программирования C++, операционная система Windows 10, среда разработки CodeBlocks, компилятор MinGW.

2.2. Реализованные функции

1) djb2

Функция djb2 вычисляет некриптостойкие хеш-функции разрядности 64 бит.

Исходный код функции djb2 находится в файле main.cpp.

Объявление функции: `unsigned long djb2(unsigned char *str)`

Тип функции: `unsigned long`.

Аргументы функции:

- `str` – указатель, в который передается содержимое файла.

Возвращаемые значения:

- `hashsum` – возвращается полученный хеш.

2) main

Функция `main` получает доступ к директории. Вычисляет хеши файлов. Ищет одинаковые файлы.

Объявление функции: `int main()`

Тип функции: `int`

Возвращаемые значения:

- 0 - успех

3) Funk

Функция `Funk` выполняет преобразование времени в строку, вычисляет текущую директорию, выполняет копирование и конкатенирование.

Объявление функции: `int Funk(char* currentFileStr)`

Тип функции: `int`

Аргументы функции:

- `currentFileStr` - указатель

Возвращаемые значения:

- 0 - успех

3. РЕЗУЛЬТАТ ТЕСТИРОВАНИЯ ПРОГРАММЫ

На рисунке 1 представлен результат работы программы, когда в директории есть одинаковые файлы

```
"C:\Users\tokar\OneDrive\Рабочий стол\CodeBlocks1361\Kurs\bin\Debug\Kurs.exe"
C:\Users\tokar\D\11.txt 11.txt 29      49018784
C:\Users\tokar\D\12.txt 12.txt 29      49018784
C:\Users\tokar\D\21.txt 21.txt 5       80260389
C:\Users\tokar\D\22.txt 22.txt 5       80260389
Same fales
hashsum = 49018784 File name: 11.txt
hashsum = 49018784 File name: 12.txt
hashsum = 80260389 File name: 21.txt
hashsum = 80260389 File name: 22.txt

Process returned 0 (0x0)  execution time : 0.024 s
Press any key to continue.
```

Рисунок 1 - результат работы программы, когда в директории есть одинаковые файлы

На рисунке 2 представлен результат работы программы, когда одинаковые файлы в директории отсутствуют.

```
"C:\Users\tokar\OneDrive\Рабочий стол\CodeBlocks1361\Kurs\bin\Debug\Kurs.exe"
C:\Users\tokar\D\11.txt 11.txt 29      49018784
C:\Users\tokar\D\13.txt 13.txt 32      53cdab75
Same fales
No duplicate files in a directory

Process returned 0 (0x0)  execution time : 0.022 s
Press any key to continue.
```

Рисунок 2 - результат работы программы, когда одинаковые файлы в директории отсутствуют

ЗАКЛЮЧЕНИЕ

В результате выполнения работы нами была разработана программа на языке C++, предназначенная для поиска одинаковых файлов в директории на основании подсчета хеш-сумм.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. https://en.wikipedia.org/wiki/List_of_hash_functions#Non-cryptographic_hash_functions
2. <http://www.cse.yorku.ca/~oz/hash.html>

ПРИЛОЖЕНИЕ 1. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Программа предназначена для нахождения одинаковых файлов в директории.

Минимальные системные требования:

- Процессор: как минимум 1 ГГц или SoC;
- ОЗУ: 1 ГБ (для 32-разрядных систем) или 2 ГБ (для 64-разрядных систем);
- Место на жестком диске: 16 ГБ (для 32-разрядных систем) или 20 ГБ (для 64-разрядных систем);
- Видеоадаптер: DirectX версии не ниже 9 с драйвером WDDM 1.0;
- Дисплей: 800 x 600;
- Операционная система: Windows 10.

ПРИЛОЖЕНИЕ 2. ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <iostream>
#include <fstream>
#include <string>
#include <io.h>
#include <ctime>
#include <dos.h>
#include <vector>
#include <algorithm>

using namespace std;

unsigned long djb2(unsigned char *str)
{
    unsigned long hashsum = 5381;
    int c;
    while (c = *str++)
        hashsum = ((hashsum << 5) + hashsum) + c;
    return hashsum;
}

#define LIM_STR 255

char szBuffer[LIM_STR];
struct _finddata_t findData;
intptr_t hashF;
size_t cnt = 0;
const char direct[] = "C:\\\\Users\\tokar\\D\\*.txt";
char currentFileStr[400];
unsigned long currentHashSum;
vector <string> namesf;
vector <unsigned long> hashsums;

int Funk(char* currentFileStr)
{
    ctime_s (szBuffer, _countof(szBuffer),
    &findData.time_write);
    char currentdirect[] = "C:\\\\Users\\tokar\\D\\";
    strcpy(currentFileStr, currentdirect);
    strcat(currentFileStr, findData.name);
    return 0;
}

int main()
```

```

{
    if ((hashF = _findfirst(direct, &findData)) == -1L)
        cout << "Error";
    else
    {
        do
        {
            string fileContents, tempStr;
            Funk(currentFileStr);
            ifstream fin(currentFileStr);
            if (fin.is_open())
            {
                while (!fin.eof())
                {
                    getline(fin, tempStr);
                    fileContents += tempStr + '\n';
                }
                fin.close();
                char
fileContents_char[fileContents.length() + 1];
                fileContents.copy(fileContents_char,
fileContents.length() + 1);
                fileContents_char[fileContents.length()]
= '\0';
                currentHashSum = djb2((unsigned char*)
fileContents_char);
                hashsums.push_back(currentHashSum);
            }
            else
            {
                cout <<"problem with opening" << endl;
            }
        }
        while (_findnext (hashF, &findData) == 0);
        _findclose(hashF);
    }
    findData.attrib = _A_SUBDIR;

    if ((hashF = _findfirst(direct, &findData)) == -1L)
    {
        cout << "Error";
    }
    else
    {
        do

```

```

        {
            Funk(currentFileStr);
            cout<< currentFileStr << "\t" << dec <<
findData.name << "\t" << findData.size << "\t" << hex <<
hashsums[cnt] << endl;
            cnt ++;
            namesf.push_back(findData.name);
        }
        while (_findnext (hashF, &findData) == 0);
        _findclose (hashF);
    }

    cout << "Same fales" << endl;
    bool Flag = false;
    int count_hashsums;
    for (size_t i = 0; i < hashsums.size(); ++ i)
    {
        count_hashsums = count(hashsums.begin(),
hashsums.end(), hashsums[i]);
        if (count_hashsums >= 2)
        {
            Flag = true;
            cout << " hashsum = " << hex << hashsums[i]
<< " File name: " << namesf[i] << endl;
        }
    }
    if (!Flag) cout << "No duplicate files in a
directory" << endl;
    return 0;
}

```