Задание 5

Хэш-функции

Темы: функции, массивы

Хэш-функция отображает переданные в нее входные данные в выходной набор бит фиксированного размера. Обычно возвращаемое хэш-функцией значение называется дайджестом, контрольной суммой хэш-значением или просто хэшем.

Хэш-функции используются для быстрого поиска в соответствующих структурах данных, в генераторах псевдослучайных чисел, хранении паролей в системах аутентификации, а также для контроля целостности переданных данных. Причем контроль целостности возможен как для определения случайных изменений во входных данных, вызванных, например, помехами при передаче (часто такие функции называют функциями вычисления контрольной суммы. Пример: CRC), так и намеренного изменения переданных данных («криптографические» хэш-функции: MD5, SHA256, SHA512, SHA3 и т.д.). Разница между функциями контрольной суммы и «криптографическими» хэш-функциями заключается в большей вычислительной сложности последних. При этом, при изменении даже одного бита исходных данных их хэш изменится лавинообразно, т.е. в идеале изменятся все байты.

Алгоритм полного перебора

Алгоритм полного перебора, также известный как brute force, осуществляет перебор (обычно последовательный) значений из некоторого множества пока не будет найдено значение, удовлетворяющее критерию поиска, либо пока не будут проверены все значения из заданного множества.

Например, для перебора всех значений, состоящих из заглавных букв латинского алфавита и цифр удобно ввести массив, содержащий используемые символы. Обычно он называется «алфавитом».

```
char Alph[] = "ABCDEFGHIJKLMOPQRYSTUVWXYZ1234567890";
```

Размер такого массива может быть вычислен с использованием оператора ***sizeof***. При этом, поскольку данный массив является строкой в стиле C, то в его конец добавляется специальное

значение – символ конца строки (‘\0’). Поэтому размер алфавита будет ***sizeof*(*Aplh*) – 1.**

Число всех возможных комбинаций из ***N*** символов из алфавита объемом ***M*** может быть вычислено как M^N.

Перебор комбинаций из ***N*** символов осуществляется с некоторого начального значения. При этом последовательно изменяются все байты, начиная с младших. Например, при ***N*** = 3 перебор для алфавита ***Alph*** начинается с комбинации AAA. Следующим проверяемым значением будет AAB, затем AAC и т.д. При переборе всех символов в младшем разряде будет выполнен инкремент значения старшего/старших разрядов, а младший разряд вернется в свое минимальное значение и цикл повторяется снова. Так, за значением AA0 следует значение ABA, далее ABB и т.д.

При решении «в лоб» для перебора всех значений заданной длины ***N*** потребуется ***N*** вложенных циклов. Однако, можно модифицировать алгоритм таким образом, чтобы ограничиться лишь двумя вложенными циклами для любого значения ***N***. При этом, для перебора всех строк с длинами от 1 до ***N*** потребуется лишь три вложенных цикла.

```
unsigned int maxIter = sizeof(Alph) - 1;

for (unsigned int strLen = 1; strLen < N; strLen++)
    //Цикл № 1

{

    ...

    maxIter *= (sizeof(Alph) - 1);
}
```

Цикл № 1 осуществляет инкремент количества рассматриваемых символов в строке. На первой итерации цикла осуществляется перебор всех значений для строки длиной 1 символ, на второй итерации – для строки длиной 2 символа и т.д. В конце каждой итерации цикла № 1 производится увеличение числа рассматриваемых комбинаций. Т.е. для строки длиной 1 символ – ***N*** комбинаций, для строки длиной 2 символа – N^2 и т.д.

Далее для строки длиной ***strLen*** символов нужно перебрать все возможные комбинации символов. Таких комбинаций будет ***maxIter*** (M^{strLen}, где ***M*** – объем алфавита). Данный перебор

осуществляется в цикле № 2. Понятно, что цикл № 2 вложен в цикл № 1.

```
for (unsigned int j = 0; j < maxIter; j++)    //Цикл № 2
{...}
```

Номер итерации в цикле № 2 однозначно определяет уникальную комбинацию символов алфавита ***Alph*** в строке заданной длины ***strLen***. При этом ***j*** = 0 для строки длиной 3 символа соответствует комбинация AAA, ***j*** = 1 – комбинация AAB и т.д.

Для определения ***k***-го символа в строке ***Result*** длиной ***strLen*** воспользуемся следующей формулой:

```
Result[k] = Alph[(j / div) % (sizeof(Alph) - 1)];
```

Значение ***div*** необходимо для учета позиции ***k***, в которой находится символ. Так, для младшего символа в строке ***div*** = 1, для каждого следующего символа ***div*** увеличивается в (**sizeof(*Alph*) - 1**) раз и т.д.

Таким образом, последний вложенный цикл (№ 3) для определения текущей проверяемой комбинации с номером ***j*** реализуется как:

```
for (int k = 0, div = 1; k < strLen; k++)     //Цикл № 3
{
    Result[k] = Alph[(j / div) % (sizeof(Alph) - 1)];
    div *= (sizeof(Alph) - 1);
}
```

Очевидно, что цикл № 3 вложен в цикл № 2. При этом, если полный перебор осуществляется для поиска комбинации по заданному критерию, то это условие проверяется после цикла № 3 внутри цикла № 2:

```
for (unsigned int j = 0; j < maxIter; j++)    // Цикл № 2
{
    for (int k = 0, div = 1; k < strLen; k++)// Цикл № 3
    {
        Result[k] = Alph[(j / div) % (sizeof(Alph) - 1)];
        div *= (sizeof(Alph) - 1);
    }
    if (...) {    // Проверка выполнения заданного критерия
        ...

    }
}
```

Задание:

Задан алфавит и хэш-функция

```
char Alph[] = "ABCDEFGHIJKLMOPQRYSTUVWXYZ1234567890";

unsigned int Hash(const char* Passwd, int Len)
{
    unsigned int res = 0;

    for (int i = 0; i < Len; i++)
    {
        res += Passwd[i];
    }

    return (res << 16) ^ (Passwd[0] << 14) + ((Passwd[Len-1]
^ Len) >> (Len >> 1));
}
```

Хэш-функция принимает на вход массив символов ***Passwd*** и его длину ***Len*** без учета символа конца строки.

Реализовать функцию **RestorePasswd**, реализующую восстановление строки символов по известному хэш-значению. Прототип функции должен быть следующим:

```
bool RestorePasswd(
    unsigned int HashValue,
    int MaxLength,
    char* Result,
    int* ResLength)
```

HashValue – заданное хэш-значение. Строку, соответствующую данному хэш значению необходимо вычислить.

MaxLength – максимальная длина строки для которой будет вычисляться. Функция осуществляет поиск среди строк с длиной от 1 до ***MaxLength*** символов.

Result – массив символов (**char**), который, в случае успеха, будет содержать строку, соответствующую заданному хэш-значению ***HashValue***.

ResultLength – длина строки (в символах), соответствующей соответствующую заданному хэш-значению ***HashValue***.

В случае если найдена строка длиной от 1 до ***MaxLength*** символов, соответствующая заданному хэш-значению, функция возвращает **true**, в противном случае – **false**.

Массив ***Result***, передаваемый в функцию **RestorePasswd** должен быть предварительно заполнен нулевыми значениями. Переменная, на которую указывает ***ResultLength*** перед вызовом функции RestorePasswd должна содержать нулевое значение.

Если функция вернула **true** – вывести найденную строку, ее длину и хэш-значение на экран, в противном случае вывести информацию, что среди строк длиной от 1 до ***MaxLength*** нет строки, соответствующей заданному хэш-значению ***HashValue***.

Пример:

```
unsigned hash = ...
char Passwd[50] = ...
int length = ...

bool res = RestorePasswd(hash, 50, Passwd, &length);

if (res == true)
{
    std::cout << "Password: " << Passwd << " Length: " << 
length << std::hex << " Hash: 0x" << hash << std::endl;
}

else
{
    std::cout << "Not cracked! " << std::endl;
}
```

Хэш-значения необходимо взять из следующей таблицы:

Студент	Хэш-значение
Авдяхин Александр	0x015A8014
Акишин Максим	0x0155C00D
Алексеев Олег	0x01510012
Амирова Анастасия	0x0158C00C
Антонов Георгий	0x015BC00C
Артамонова Арина	0x0145C00F
Бабурин Сергей	0x0161000D
Баянов Тамир	0x0190C015
Беренштейн Александр	0x0140800D
Голыгин Игорь	0x014A000D
Джомардидзе Владимир	0x0141C00D
Дубинин Максим	0x01204015
Дубровский Денис	0x0170C011
Ефремов Олег	0x014CC00C
Кисиев Денис	0x0171C00F
Кудрин Фёдор	0x019AC012
Курлыков Данила	0x0155400D
Латифов Шерзод	0x0175C010
Литвинов Данила	0x01728011
Лю Хуэйлун	0x01928015
Монастырская Софья	0x01854012
Мороз Павел	0x0152400D
Разумкина Варвара	0x014FC00C
Сергеев Кирилл	0x01870012
Терекбаев Иван	0x0164C015
Тыгин Владислав	0x01644012
Урусов Вячеслав	0x01948014
Филиппов Андрей	0x016CC012
Шарган Иван	0x019C8015

Вопросы для самопроверки:

1. Может ли различным наборам исходных данных соответствовать одинаковое значение хэша?
2. Каково множество входных значений функции?
3. Каково множество выходных значений функции?
4. Приведите примеры использования хэш-функций в ПО, которым Вы пользуетесь.
5. Является ли хэш-функция взаимнооднозначной?
6. Если в цикле № 2 начать перебор в обратном направлении, то будет ли получено такое же значение строки, как при проходе в прямом направлении?
7. Чему равно общее число комбинаций при полном переборе всех строк с длиной от 1 до 10 с использованием введенного ранее алфавита ***Alph***?