Код программы:

#include <iostream>

#include <string>

using namespace std;

namespace hash

{

class Hash

{

string hash;

int receivingExistCodes(int x);

int getControlSum(string str);

public:

string getHash(string userString, unsigned int lenghtHash);

};

int Hash::receivingExistCodes(int x)

{

while (!(((x <= 57) && (x >= 48)) || ((x <= 90) && (x >= 65)) || ((x <= 122) && (x >= 97)))) {

if (x < 48) { x += 24; }

else { x -= 47; };

}

return x;

}

int Hash::getControlSum(string str)

{

unsigned int sault = 0, strlen = 0;

for (; strlen < str.size(); strlen++)

sault += int(str[strlen]);

return sault;

}

string Hash::getHash(string userString, unsigned int lenghtHash)

{

if (lenghtHash > 3)

{

//минимальная длина строки хеша кратная 2

unsigned int minLen = 2;

//длина строки ближайшая к нужной длине хеша

unsigned int realMinLen = 0;

//соль - 1

unsigned int originalSault = this->getControlSum(userString);

unsigned int originalLenghtStr = (userString.size());

while (minLen <= lenghtHash)

realMinLen = (minLen \*= 2);

while (minLen < originalLenghtStr)

minLen \*= 2;

if ((minLen - originalLenghtStr) < minLen)

minLen \*= 2;

// кол-во символов необходимо добавить

int addCount = minLen - originalLenghtStr;

//добавление (первичный макс хеш)

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

userString += this->receivingExistCodes(userString[i] + userString[i + 1]);

int maxSault = this->getControlSum(userString);

int maxLenghtStr = (userString.size());

// определение степени сжатия (и собственно сжатие)

while (userString.size() != realMinLen)

{

for (int i = 0, center = userString.size() / 2; i < center; i++)

this->hash += this->receivingExistCodes(userString[center - i] + userString[center + i]);

userString = this->hash;

this->hash.clear();

}

//приведение к заданной длине

unsigned int rem = realMinLen - lenghtHash;

for (unsigned int i = 0, countCompress = realMinLen / rem; this->hash.size() < (lenghtHash - 4); i++)

{

if (i % countCompress == 0)

this->hash += this->receivingExistCodes(userString[i] + userString[++i]);

else

this->hash += userString[i];

}

//соль из оригинальной

this->hash += this->receivingExistCodes(originalSault);

this->hash += this->receivingExistCodes(originalLenghtStr);

//соль из максимальной

this->hash += this->receivingExistCodes(maxSault);

this->hash += this->receivingExistCodes(maxLenghtStr);

return this->hash;

}

return "";

}

}

using namespace hash;

int nonce(int nonce, string str, int len)

{

while (true)

{

Hash hash;

string res;

string str2;

str2 = to\_string(nonce);

res = hash.getHash(str + str2, len);

int len\_2 = sizeof(res);

cout << "nonce: " << nonce << ";" << " " << "Result of hash: " << res << endl;

if (int(res[0]) == 48)

{

cout << "Needed nonce: " << nonce;

break;

}

nonce++;

}

return nonce;

}

int main() {

const string str = "hello";

int len = 20;

int res;

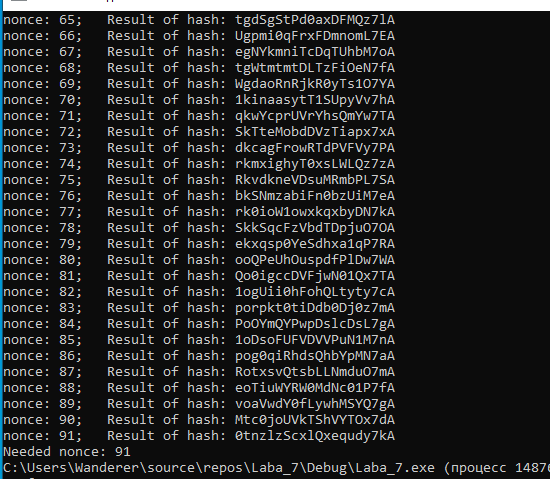
int i = 0;

res = nonce(i,str,len);

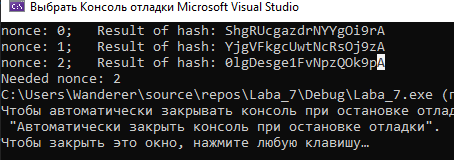
return 0;

}

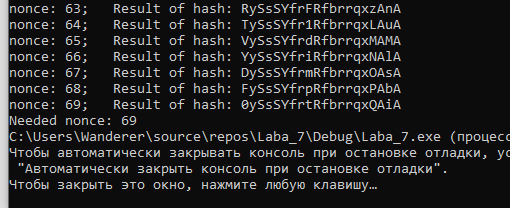
При хэшировании слова - hello



При хэшировании слова - wanderer



При хэшировании слова – brave new world



Из -за особенностей хэш функции генирируется только один “0”;