МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

**ИНТЕЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ**

ОТЧЁТ

По лабораторной работе № \_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Выполнил:

Студент группы ИИ-22

Копанчук Евгений Романович

Проверил\_\_:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Брест – 2023

Ход работы

*Код программы*

#pragma once

#include <iostream>

#include <bitset>

#include <sstream>

using namespace std;

typedef unsigned long long int int64;

int64 rotr(int64 x, int n);

int64 shr(int64 x, int n);

string gethex(string bin);

string dec2hex(int64 deci);

int64 bin2dec(string bin);

class SHA512

{

int64 \*Message;

unsigned long long k[80] = {

0x428a2f98d728ae22, 0x7137449123ef65cd, 0xb5c0fbcfec4d3b2f, 0xe9b5dba58189dbbc,

0x3956c25bf348b538, 0x59f111f1b605d019, 0x923f82a4af194f9b, 0xab1c5ed5da6d8118,

0xd807aa98a3030242, 0x12835b0145706fbe, 0x243185be4ee4b28c, 0x550c7dc3d5ffb4e2,

0x72be5d74f27b896f, 0x80deb1fe3b1696b1, 0x9bdc06a725c71235, 0xc19bf174cf692694,

0xe49b69c19ef14ad2, 0xefbe4786384f25e3, 0x0fc19dc68b8cd5b5, 0x240ca1cc77ac9c65,

0x2de92c6f592b0275, 0x4a7484aa6ea6e483, 0x5cb0a9dcbd41fbd4, 0x76f988da831153b5,

0x983e5152ee66dfab, 0xa831c66d2db43210, 0xb00327c898fb213f, 0xbf597fc7beef0ee4,

0xc6e00bf33da88fc2, 0xd5a79147930aa725, 0x06ca6351e003826f, 0x142929670a0e6e70,

0x27b70a8546d22ffc, 0x2e1b21385c26c926, 0x4d2c6dfc5ac42aed, 0x53380d139d95b3df,

0x650a73548baf63de, 0x766a0abb3c77b2a8, 0x81c2c92e47edaee6, 0x92722c851482353b,

0xa2bfe8a14cf10364, 0xa81a664bbc423001, 0xc24b8b70d0f89791, 0xc76c51a30654be30,

0xd192e819d6ef5218, 0xd69906245565a910, 0xf40e35855771202a, 0x106aa07032bbd1b8,

0x19a4c116b8d2d0c8, 0x1e376c085141ab53, 0x2748774cdf8eeb99, 0x34b0bcb5e19b48a8,

0x391c0cb3c5c95a63, 0x4ed8aa4ae3418acb, 0x5b9cca4f7763e373, 0x682e6ff3d6b2b8a3,

0x748f82ee5defb2fc, 0x78a5636f43172f60, 0x84c87814a1f0ab72, 0x8cc702081a6439ec,

0x90befffa23631e28, 0xa4506cebde82bde9, 0xbef9a3f7b2c67915, 0xc67178f2e372532b,

0xca273eceea26619c, 0xd186b8c721c0c207, 0xeada7dd6cde0eb1e, 0xf57d4f7fee6ed178,

0x06f067aa72176fba, 0x0a637dc5a2c898a6, 0x113f9804bef90dae, 0x1b710b35131c471b,

0x28db77f523047d84, 0x32caab7b40c72493, 0x3c9ebe0a15c9bebc, 0x431d67c49c100d4c,

0x4cc5d4becb3e42b6, 0x597f299cfc657e2a, 0x5fcb6fab3ad6faec, 0x6c44198c4a475817

};

void Func(int64 a, int64 b, int64 c, int64& d, int64 e, int64 f, int64 g, int64& h, int64 M, int64 K);

void separator(string getBlock);

public:

string hash(string message);

};

#include "SHA512.h"

int64 rotr(int64 x, int n) {

return (x >> n) | (x << (64 - n));

}

int64 shr(int64 x, int n) {

return (x >> n);

}

string gethex(string bin) {

if (bin == "0000") return "0"; if (bin == "0001") return "1";

if (bin == "0010") return "2"; if (bin == "0011") return "3";

if (bin == "0100") return "4"; if (bin == "0101") return "5";

if (bin == "0110") return "6"; if (bin == "0111") return "7";

if (bin == "1000") return "8"; if (bin == "1001") return "9";

if (bin == "1010") return "a"; if (bin == "1011") return "b";

if (bin == "1100") return "c"; if (bin == "1101") return "d";

if (bin == "1110") return "e"; if (bin == "1111") return "f";

}

string dec2hex(int64 deci) {

string EQBIN = bitset<64>(deci).to\_string();

string hexstring = "";

string temp;

for (unsigned int i = 0;

i < EQBIN.length(); i += 4) {

temp = EQBIN.substr(i, 4);

hexstring += gethex(temp);

}

return hexstring;

}

int64 bin2dec(string bin) {

int64 value = bitset<64>(bin).to\_ullong();

return value;

}

void SHA512::separator(string getBlock) {

int chunknum = 0;

for (unsigned int i = 0;

i < getBlock.length();

i += 64, ++chunknum) {

this->Message[chunknum] = bin2dec(getBlock.substr(i, 64));

}

for (int g = 16; g < 80; ++g) {

int64 WordA

= rotr(this->Message[g - 2], 19)

^ rotr(this->Message[g - 2], 61)

^ shr(this->Message[g - 2], 6);

int64 WordB = this->Message[g - 7];

int64 WordC

= rotr(this->Message[g - 15], 1)

^ rotr(this->Message[g - 15], 8)

^ shr(this->Message[g - 15], 7);

int64 WordD = this->Message[g - 16];

int64 T = WordA + WordB + WordC + WordD;

this->Message[g] = T;

}

}

void SHA512::Func(

int64 a, int64 b,

int64 c, int64& d,

int64 e, int64 f,

int64 g, int64& h,

int64 M, int64 K)

{

int64 Ch = (e & f) ^ (~e & g);

int64 sigmaE = rotr(e, 14) ^ rotr(e, 18) ^ rotr(e, 41);

int64 sigmaA = rotr(a, 28) ^ rotr(a, 34) ^ rotr(a, 39);

int64 maj = (a & b) ^ (b & c) ^ (c & a);

int64 T1 = h + Ch + sigmaE + M + K;

int64 T2 = sigmaA + maj;

d = d + T1;

h = T1 + T2;

}

string SHA512::hash(string myString) {

this->Message = new int64[80];

int64 A = 0x6a09e667f3bcc908; int64 B = 0xbb67ae8584caa73b;

int64 C = 0x3c6ef372fe94f82b; int64 D = 0xa54ff53a5f1d36f1;

int64 E = 0x510e527fade682d1; int64 F = 0x9b05688c2b3e6c1f;

int64 G = 0x1f83d9abfb41bd6b; int64 H = 0x5be0cd19137e2179;

int64 AA, BB, CC, DD, EE, FF, GG, HH;

stringstream fixedstream;

for (int i = 0;

i < myString.size(); ++i) {

fixedstream << bitset<8>(myString[i]);

}

string s1024;

s1024 = fixedstream.str();

int orilen = s1024.length();

int tobeadded;

int modded = s1024.length() % 1024;

if (1024 - modded >= 128) {

tobeadded = 1024 - modded;

}

else if (1024 - modded < 128) {

tobeadded = 2048 - modded;

}

s1024 += "1";

for (int y = 0; y < tobeadded - 129; y++) {

s1024 += "0";

}

string lengthbits = std::bitset<128>(orilen).to\_string();

s1024 += lengthbits;

int blocksnumber = s1024.length() / 1024;

int chunknum = 0;

string\* Blocks = new string[blocksnumber];

for (int i = 0; i < s1024.length();

i += 1024, ++chunknum) {

Blocks[chunknum] = s1024.substr(i, 1024);

}

for (int letsgo = 0; letsgo < blocksnumber; ++letsgo) {

separator(Blocks[letsgo]);

AA = A; BB = B; CC = C; DD = D; EE = E; FF = F; GG = G; HH = H;

int j = 0;

for (int i = 0; i < 10; i++) {

Func(A, B, C, D, E, F, G, H, this->Message[j], this->k[j]); j++;

Func(H, A, B, C, D, E, F, G, this->Message[j], this->k[j]); j++;

Func(G, H, A, B, C, D, E, F, this->Message[j], this->k[j]); j++;

Func(F, G, H, A, B, C, D, E, this->Message[j], this->k[j]); j++;

Func(E, F, G, H, A, B, C, D, this->Message[j], this->k[j]); j++;

Func(D, E, F, G, H, A, B, C, this->Message[j], this->k[j]); j++;

Func(C, D, E, F, G, H, A, B, this->Message[j], this->k[j]); j++;

Func(B, C, D, E, F, G, H, A, this->Message[j], this->k[j]); j++;

}

A += AA; B += BB; C += CC; D += DD; E += EE; F += FF; G += GG; H += HH;

}

delete[] this->Message;

stringstream output;

output << dec2hex(A); output << dec2hex(B);

output << dec2hex(C); output << dec2hex(D);

output << dec2hex(E); output << dec2hex(F);

output << dec2hex(G); output << dec2hex(H);

return output.str();

}

#include "SHA512.h"

int main() {

SHA512 hasher = SHA512();

string S = "hello world";

cout << S << ": " << hasher.hash(S) << endl;

}

**