МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

**КАФЕДРА ИИТ**

ОТЧЁТ

по лабораторной работе №2

**«Unit-тестирование»**

Выполнил:

Студент 1 курса

группы ПО-9

Харитонович Захар Сергеевич

Проверил:

Скарубо А. О.

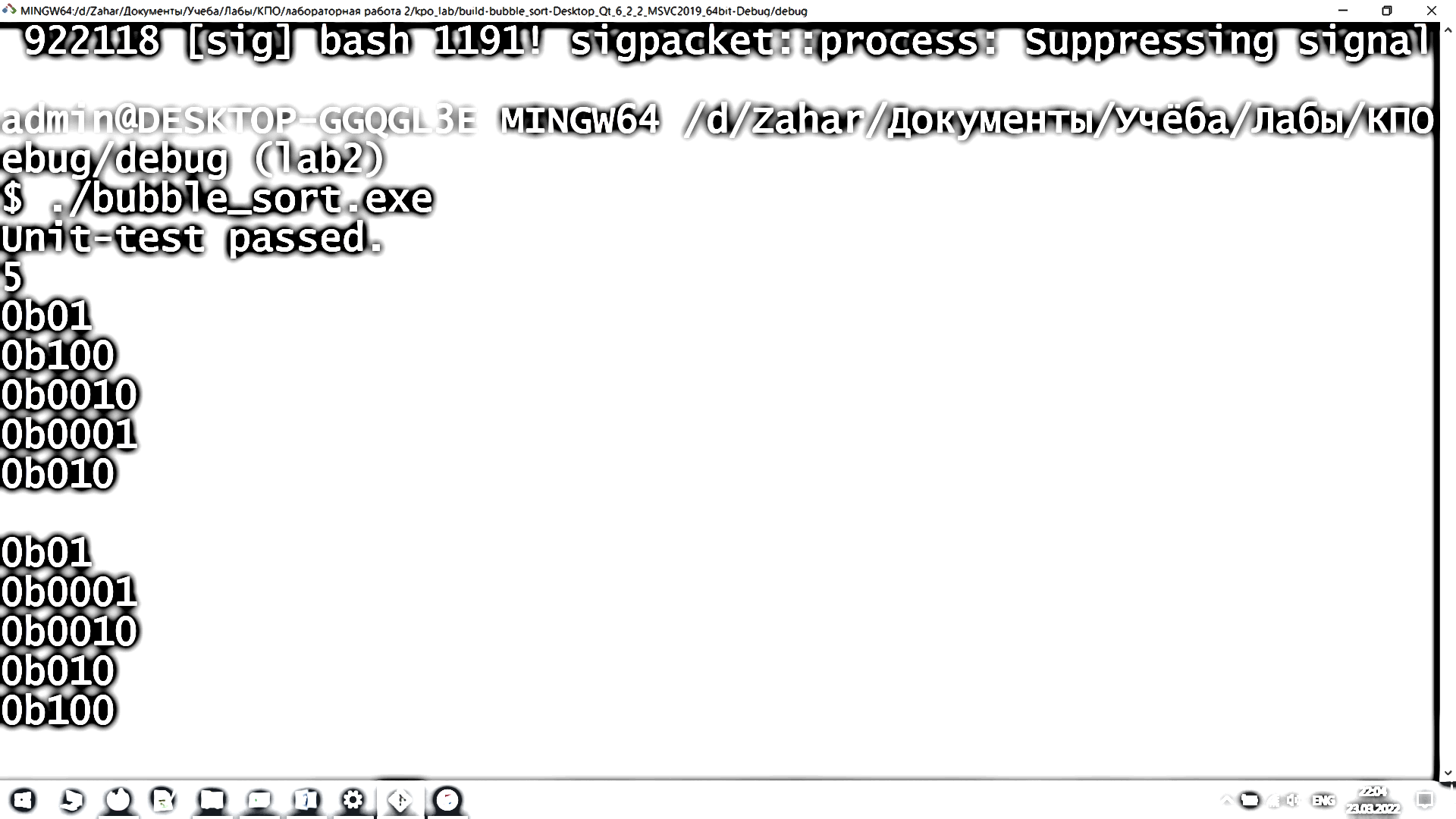
Брест 2022

**Порядок выполнения работы**

*Задание 1*

Реализовать сортировку пузырьком для чисел в двоичной системе счисления. Реализовать 5 unit-тестов.

**Код программы: Пример исполнения:**

#include <iostream>

#include <cassert>

using namespace std;

int toTen(string);

void binSort(string \*, size\_t);

void unitTest();

bool compare(string \*, string \*);

int main() {

unitTest();

size\_t aSize; cin >> aSize;

string \*A = new string[aSize];

for (size\_t i = 0; i < aSize; i++) cin >> A[i];

cout << endl;

binSort(A, aSize);

for (size\_t i = 0; i < aSize; i++) cout << A[i] << endl;

return 0;

}

bool compare(string \* A, string \* B, size\_t arrSize) {

for (size\_t i = 0; i < arrSize; i++)

if(toTen(A[i]) != toTen(B[i])) return 0;

return 1;

}

int toTen(string binNum) {

int tenNum = 0;

if (!(binNum[0] == '0' && binNum[1] == 'b')) {

cout << "Not a binary num." << endl;

return 0;

}

for (size\_t i = binNum.length() - 1; i > 1; i--)

if (binNum[i] == '1')

tenNum += (1 << (binNum.length() - i - 1));

return tenNum;

}

void binSort(string \*A, size\_t arraySize) {

for (size\_t i = 0; i < arraySize - 1; i++)

for (size\_t j = i; j < arraySize - 1; j++)

if (toTen(A[j]) > toTen(A[j+1])) swap(A[j], A[j+1]);

}

void unitTest() {

string A1[] = {"0b01", "0b00010", "0b000"}, B1[] = {"0b0", "0b1", "0b10"};

binSort(A1, 3); assert(compare(A1, B1, 3));

string A2[] = {"0b1101", "0b011010", "0b010", "0b0001", "0b0111"}, B2[] = {"0b1", "0b10", "0b111", "0b1101", "0b11010"};

binSort(A2, 5); assert(compare(A2, B2, 5));

string A3[] = {"0b01", "0b10", "0b11", "0b0"}, B3[] = {"0b0", "0b1", "0b10", "0b11"};

binSort(A3, 4); assert(compare(A3, B3, 4));

string A4[] = {"0b11", "0b00010"}, B4[] = {"0b10", "0b11"};

binSort(A4, 2); assert(compare(A4, B4, 2));

string A5[] = {"0b01", "0b0", "0b11"}, B5[] = {"0b0", "0b1", "0b11"};

binSort(A5, 3); assert(compare(A5, B5, 3));

cout << "Unit-test passed.\n";

}

*Задание 2*

Реализовать программу поиска n-ого числа Фибоначчи. Написать 2 версии программы: с рекурсией и без. Провести сравнение времени выполнения этих реализаций.

**Код программы: Пример исполнения:**

#include <iostream>

#include <ctime>

#include <cassert>

using namespace std;

unsigned long long fibRec(int);

unsigned long long fibNoRec(int);

void unitTest();

int main() {

unitTest();

int n = 0; cout << "n = "; cin >> n;

clock\_t time; time = clock();

cout << "Fibonacci with recursion: " << fibRec(n) << endl;

cout << "Time: " << clock() - time << " ms" << endl;

time = clock();

cout << "Fibonacci without recursion: " << fibNoRec(n) << endl;

cout << "Time: " << clock() - time << " ms" << endl;

return 0;

}

void unitTest() {

assert(fibRec(5) == 5);

assert(fibRec(10) == 55);

assert(fibRec(13) == 233);

assert(fibRec(20) == 6765);

assert(fibRec(25) == 75025);

assert(fibNoRec(5) == 5);

assert(fibNoRec(10) == 55);

assert(fibNoRec(13) == 233);

assert(fibNoRec(20) == 6765);

assert(fibNoRec(25) == 75025);

cout << "Unit-tests passed.\n";

}

unsigned long long fibRec(int n) {

if(n <= 0) return 0;

if(n <= 2) return 1;

return fibRec(n - 1) + fibRec(n - 2);

}

unsigned long long fibNoRec(int n) {

long temp1 = 1, temp2 = 1, temp3 = 1;

for(size\_t i = 3; i <= n; i++) {

temp3 = temp1 + temp2;

temp1 = temp2;

temp2 = temp3;

}

return temp3;

}

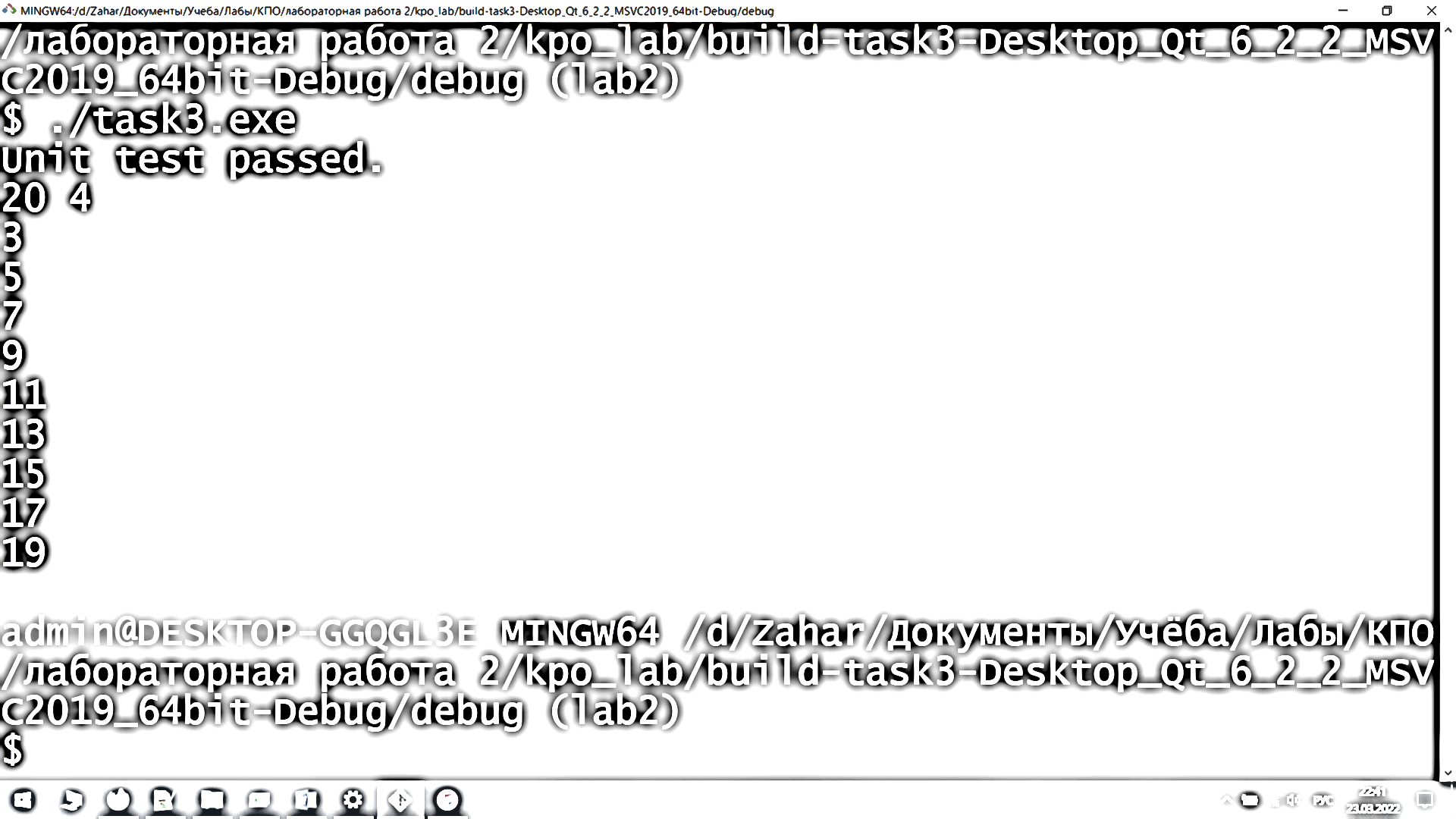
*Вариант 32*

*Задание 3*

Даны целые числа n и m. Получить все натуральные числа, меньшие n и взаимно простые с m.

**Код программы: Пример исполнения:**

#include <iostream>

#include <cassert>

using namespace std;

int gcd(int, int);

int \* coprime(int, int, size\_t &);

void unitTest();

bool compare(int \*, int \*, size\_t);

int main() {

unitTest();

int n, m; cin >> n >> m;

if (n <= 0 || m <= 0) {

cout << "Not a natural number." << endl;

return 1;

}

size\_t arrSize = 0;

int \* array = coprime(n, m, arrSize);

for (size\_t i = 0; i < arrSize; i++) cout << array[i] << endl;

return 0;

}

int gcd(int a, int b) {

while (a && b) if(a > b) a %= b; else b %= a;

return a + b;

}

int \* coprime(int n, int m, size\_t &arrSize) {

arrSize = 0;

int \* out = new int[n];

for (int i = 2; i <= n; i++)

if (gcd(i, m) == 1) {

out[arrSize] = i;

arrSize++;

}

return out;

}

bool compare(int \*A, int \*B, size\_t arrSize) {

for (size\_t i = 0; i < arrSize; i++) if(A[i] != B[i]) return 0;

return 1;

}

void unitTest() {

int \* A, B1[] = {2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14, 16, 17, 18, 19};

size\_t arrSize;

A = coprime(20, 5, arrSize); assert(compare(A, B1, arrSize));

int B2[] = {3, 5, 7, 9, 11, 13, 15};

A = coprime(15, 4, arrSize); assert(compare(A, B2, arrSize));

int B3[] = {2, 4, 5, 7, 8, 10, 11, 13, 14, 16};

A = coprime(16, 3, arrSize); assert(compare(A, B3, arrSize));

int B4[] = {2, 3, 4, 5, 6, 8, 9, 10};

A = coprime(10, 7, arrSize); assert(compare(A, B4, arrSize));

int B5[] = {2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17, 19, 20, 22, 23};

A = coprime(23, 3, arrSize); assert(compare(A, B5, arrSize));

cout << "Unit test passed.\n";

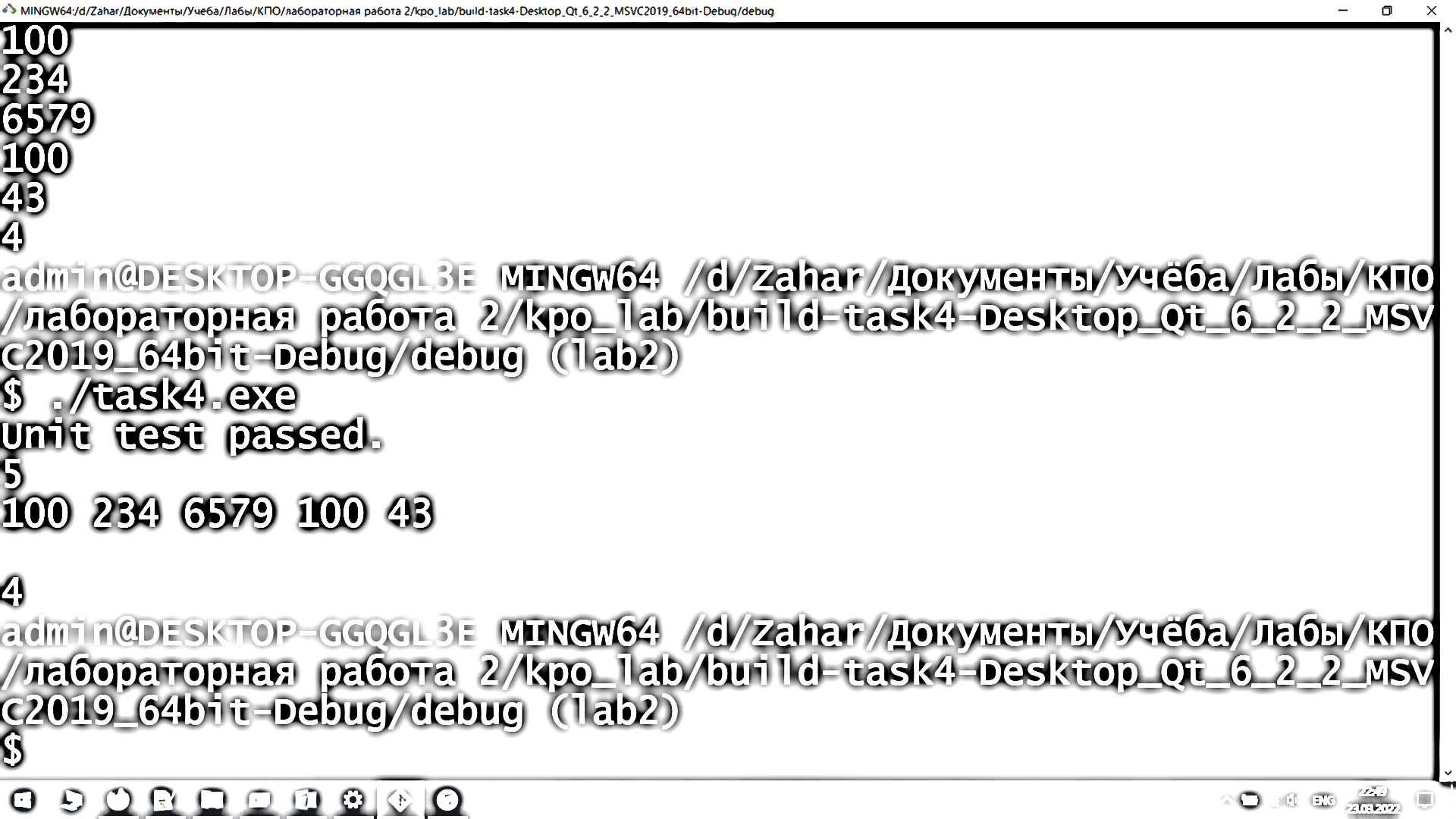
}

*Задание 4*

Даны натуральное число n и целые числа a1, a2, …, an. Найти номер последнего числа, равного 100. Допустить, что чисел, равных 100, среди заданных может не быть.

**Код программы: Пример исполнения:**

#include <iostream>

#include <cassert>

using namespace std;

void unitTest();

int lastHundred(int , int \*);

int main() {

unitTest();

int n;

cin >> n;

int \*a = new int[n];

for (size\_t i = 0; i < n; i++) {

cin >> a[i];

}

cout << endl << lastHundred(n, a);

return 0;

}

int lastHundred(int n, int \* a) {

int last = 0;

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

if (a[i] == 100) {

last = i + 1;

}

}

return last;

}

void unitTest() {

int A1[] = {100, 54, 23, 100};

assert(lastHundred(4, A1) == 4);

int A2[] = {432, 123, 875};

assert(lastHundred(3, A2) == 0);

int A3[] = {100, 100, 123, 543, 25};

assert(lastHundred(5, A3) == 2);

int A4[] = {100, 54, 236, 100, 100, 100, 1};

assert(lastHundred(7, A4) == 6);

int A5[] = {867, 54, 100, 23, 10};

assert(lastHundred(5, A5) == 3);

cout << "Unit test passed.\n";

}