**Министерство науки и высшего образования Российской Федерации**Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Владимирский государственный университет**

**имени Александра Григорьевича и Николая Григорьевича Столетовых»**

**(ВлГУ)**

Институт прикладной математики, физики и информатики

Кафедра физики и прикладной математики

**ЛАБОРАТОРНАЯ РАБОТА №2**

по дисциплине  
«Основы программирования»  
на тему:  
**«Основные алгоритмические структуры»**

Выполнил:

ст. гр. ПМИ-123

Рушев А.М.

Принял:  
ст. преподаватель

каф. ФиПМ

Черников А.С.

Владимир, 2023 г.

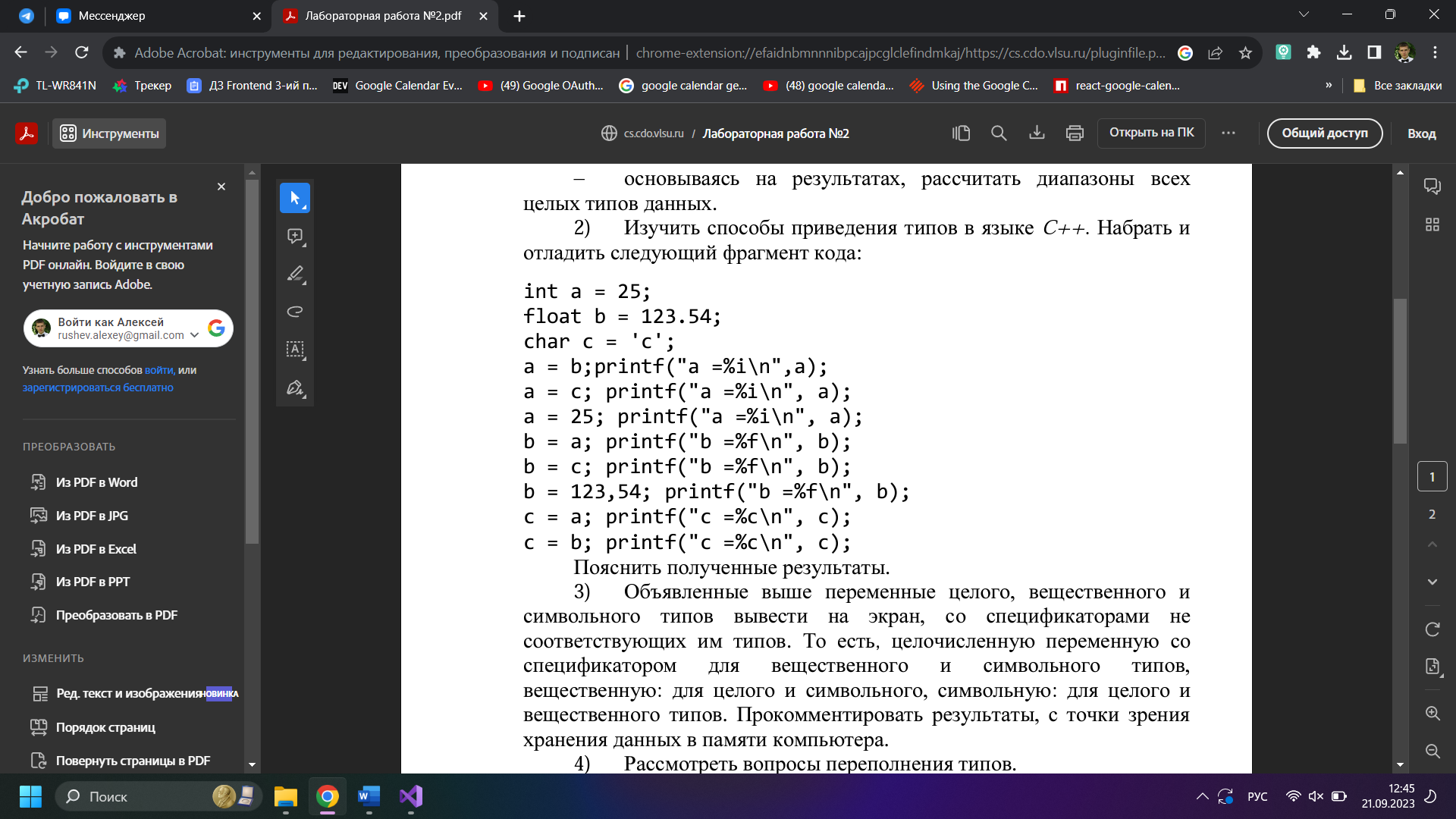
**Цель работы:** изучить базовые типы данных языка С++, особенности выполнения операций на этих типах данных, приоритет операций.

**Задание.**

1. Определить количество памяти, отводимое под переменные базовых типов.

* объявить переменные всех базовых типов, в том числе переменную типа void, убедиться в том, что такое объявление невозможно, после чего закомментировать соответствующую строку, проинициализировать объявленные переменные допустимыми значениями, вывести на экран с соответствующими спецификаторами.
* определить количество памяти, отводимое под переменные базовых типов, используя sizeof().
* основываясь на результатах, рассчитать диапазоны всех целых типов данных.

1. Изучить способы приведения типов в языке C++. Набрать и отладить следующий фрагмент кода:



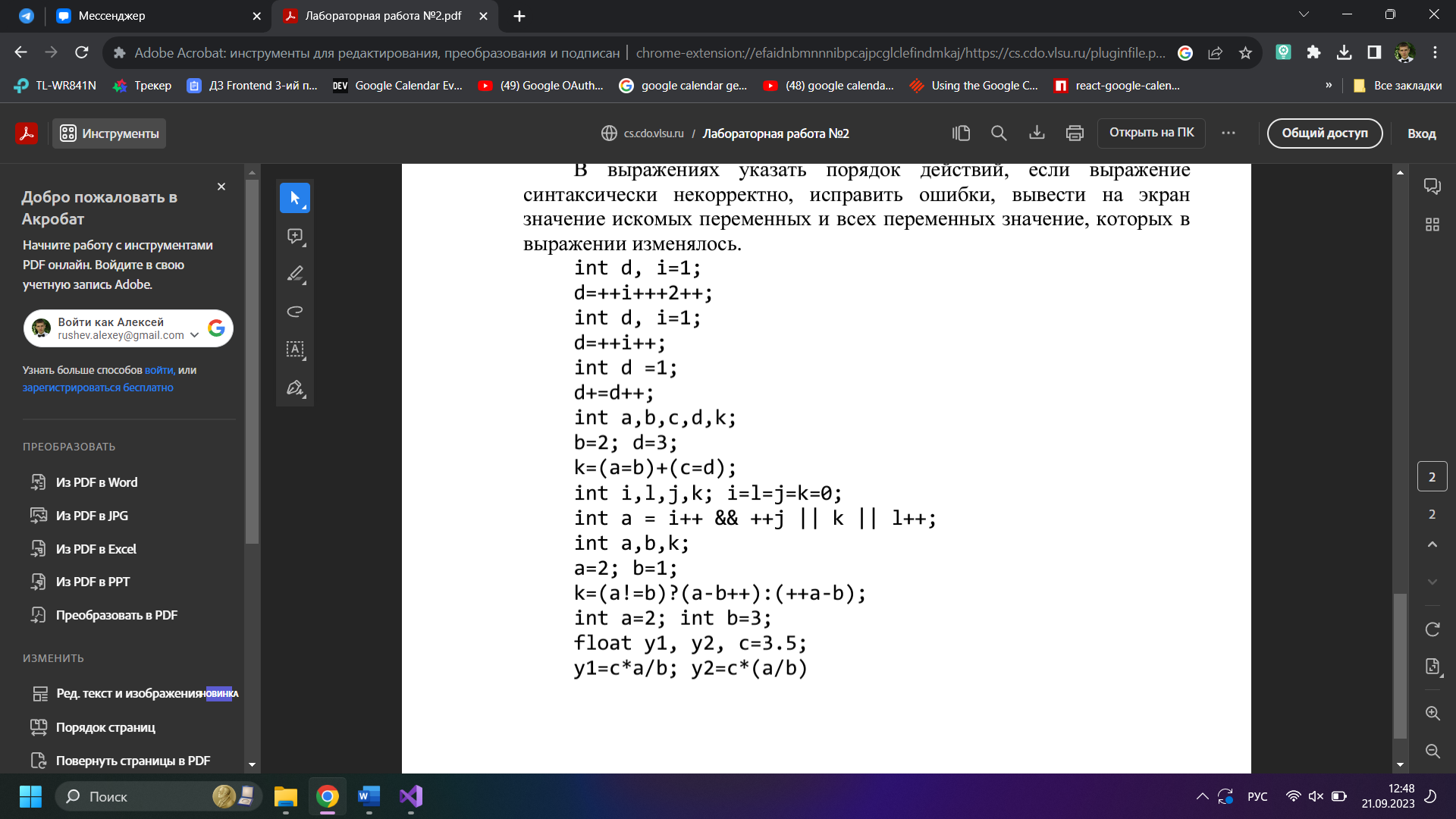
Пояснить полученные результаты.

1. Объявленные выше переменные целого, вещественного и символьного типов вывести на экран, со спецификаторами не соответствующих им типов. То есть, целочисленную переменную со спецификатором для вещественного и символьного типов, вещественную: для целого и символьного, символьную: для целого и вещественного типов. Прокомментировать результаты, с точки зрения хранения данных в памяти компьютера.
2. Рассмотреть вопросы переполнения типов.

* Объявить две символьные переменные, проинициализировать их значениями 65 и 274, вывести значения каждой переменной со спецификаторами для целого и символьного типов.
* Объявить две переменные следующих типов: signed char и unsigned char, проинициализировать их значением 0, вывести значения переменных на экран со спецификаторами для целого знакового и символьного типов. Задать значение объявленных выше переменных равным 255, вывести на экран с указанными выше спецификаторами, увеличить значение каждой переменной на два, вывести на экран изменённые значения со спецификаторами для целого знакового и символьного типов, проанализировать результаты.
* Объявить две целочисленные переменные, знаковую и беззнаковую, проинициализировать их максимальным значением для соответствующего типа, увеличить значения на три, вывести значение каждой переменной на экран дважды, со спецификаторами для знакового и беззнакового типов, объяснить полученные результаты.

1. Приоритет операций.

В выражениях указать порядок действий, если выражение синтаксически некорректно, исправить ошибки, вывести на экран значение искомых переменных и всех переменных значение, которых в выражении изменялось.



**Практическая часть:**

**Задание 1.**

Определить количество памяти, отводимое под переменные базовых типов.

Листинг приложения:

void task1() {

cout << "Задание 1." << endl;

// знаковые типы данных (signed)

short thin\_number;

int integer;

long long\_integer;

long long very\_long\_integer;

float number;

double long\_number;

bool boolean;

char character;

// беззнаковые типы данных (unsigned)

unsigned short unsigned\_thin\_number;

unsigned int unsigned\_integer;

unsigned long unsigned\_long\_integer;

unsigned char unsigned\_character;

// переменную с типом данных void объявить нельзя, так как это не имеет смысла

// void nothing;

cout << "short: " << sizeof(thin\_number) << " байт" << endl;

cout << "int: " << sizeof(integer) << " байт" << endl;

cout << "long: " << sizeof(long\_integer) << " байт" << endl;

cout << "long long: " << sizeof(very\_long\_integer) << " байт" << endl;

cout << "float: " << sizeof(number) << " байт" << endl;

cout << "double: " << sizeof(long\_number) << " байт" << endl;

cout << "bool: " << sizeof(boolean) << " байт" << endl;

cout << "char: " << sizeof(character) << " байт" << endl;

}

Результат выполнения кода:

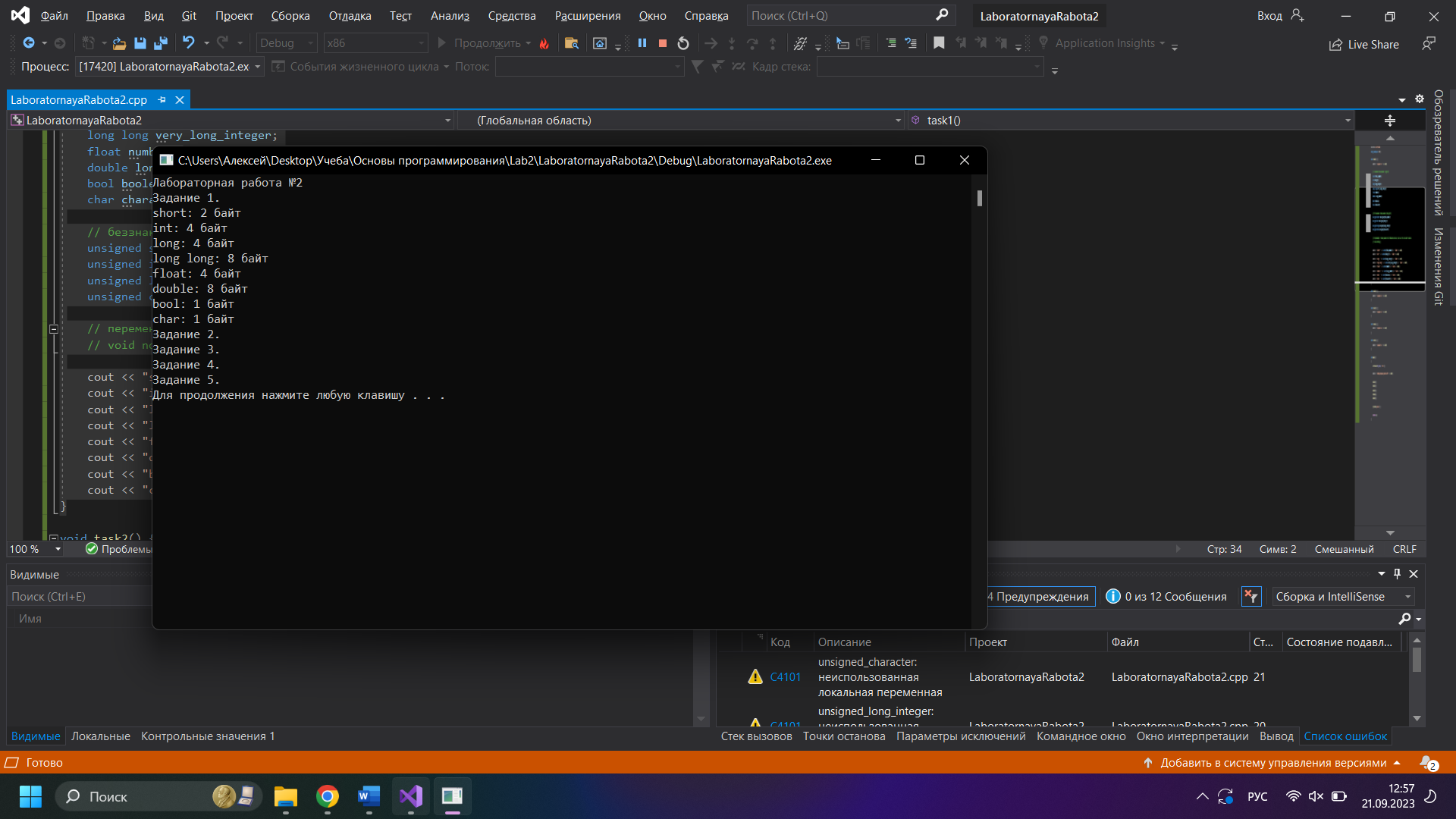


Рисунок 1 – Результат выполнения кода задания 1

Основываясь на полученном результате, мы сможем рассчитать диапазоны для всех вышеперечисленных типов данных:

Листинг кода:

void task1\_2() {

cout << "\nЗадание 1." << endl;

//int от −2 147 483 648 до 2 147 483 647

int a = INT\_MIN;

int b = INT\_MAX;

printf("int от %d до %d\n", a, b);

//char от -128 до 127

char c = CHAR\_MIN;

char d = CHAR\_MAX;

printf("char от %d до %d\n", c, d);

//short от –32768 до 32767

short e = SHRT\_MIN;

short f = SHRT\_MAX;

printf("short от %d до %d\n", e, f);

//long от −2 147 483 648 до 2 147 483 647

long int g = LONG\_MIN;

long int h = LONG\_MAX;

printf("long int от %ld до %ld\n", g, h);

//short от –32768 до 32767

short int k = SHRT\_MIN;

short int l = SHRT\_MAX;

printf("short int от %d до %d\n", k, l);

//unsigned int от 0 до 4 294 967 295

unsigned int s = UINT\_MAX;

printf("unsigned int от 0 до %u\n", s);

//unsigned short от 0 до 65535

unsigned short us = USHRT\_MAX;

printf("unsigned short от 0 до %u\n", us);

//unsigned long от 0 до 4 294 967 295

unsigned long ul = ULONG\_MAX;

printf("unsigned long от 0 до %u\n", ul);

//long long от −9 223 372 036 854 775 808 до +9 223 372 036 854 775 807

long long int ll\_min = LLONG\_MIN;

long long int ll\_max = LLONG\_MAX;

printf("long long от %lld до %lld\n", ll\_min, ll\_max);

//unsigned long long от 0 до 18 446 744 073 709 551 615

unsigned long long int ull\_min = ULLONG\_MAX;

printf("unsigned long long от 0 до %llu\n", ull\_min);

//float от +/- 3.4E-38 до 3.4E+38

float fl\_mn = FLT\_MIN;

float fl\_mx = FLT\_MAX;

printf("float от %e до %e\n", fl\_mn, fl\_mx);

//double от +/- 1.7E-308 до 1.7E+308

double dbl\_min = DBL\_MIN;

double dbl\_max = DBL\_MAX;

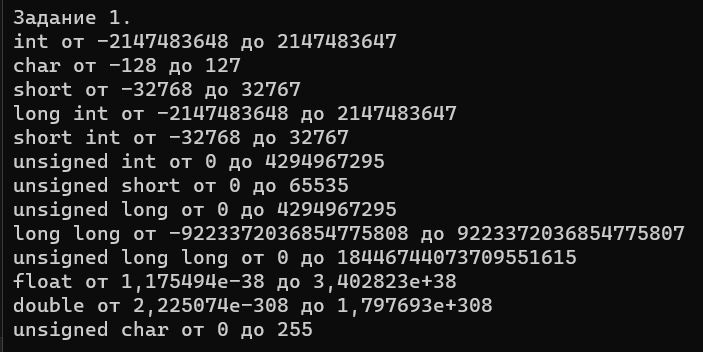
printf("double от %e до %e\n", dbl\_min, dbl\_max);

//unsigned char от 0 до 255

unsigned char uc = UCHAR\_MAX;

printf("unsigned char от 0 до %d\n", uc);

}



**Задание 2.**

Изучить способы приведения типов в языке C++.

Листинг приложения.

void task2() {

cout << "\nЗадание 2." << endl;

int a = 25;

float b = 123.54;

char c = 'c';

a = b;

printf("a = %i\n", a);

a = c;

printf("a = %i\n", a);

a = 25;

printf("a = %i\n", a);

b = a;

printf("b = %f\n", b);

b = c;

printf("b = %f\n", b);

b = 123.54;

printf("b = %f\n", b);

c = a;

printf("c = %c\n", c);

c = b;

printf("c = %c\n", c);

}

Результат выполнения кода:

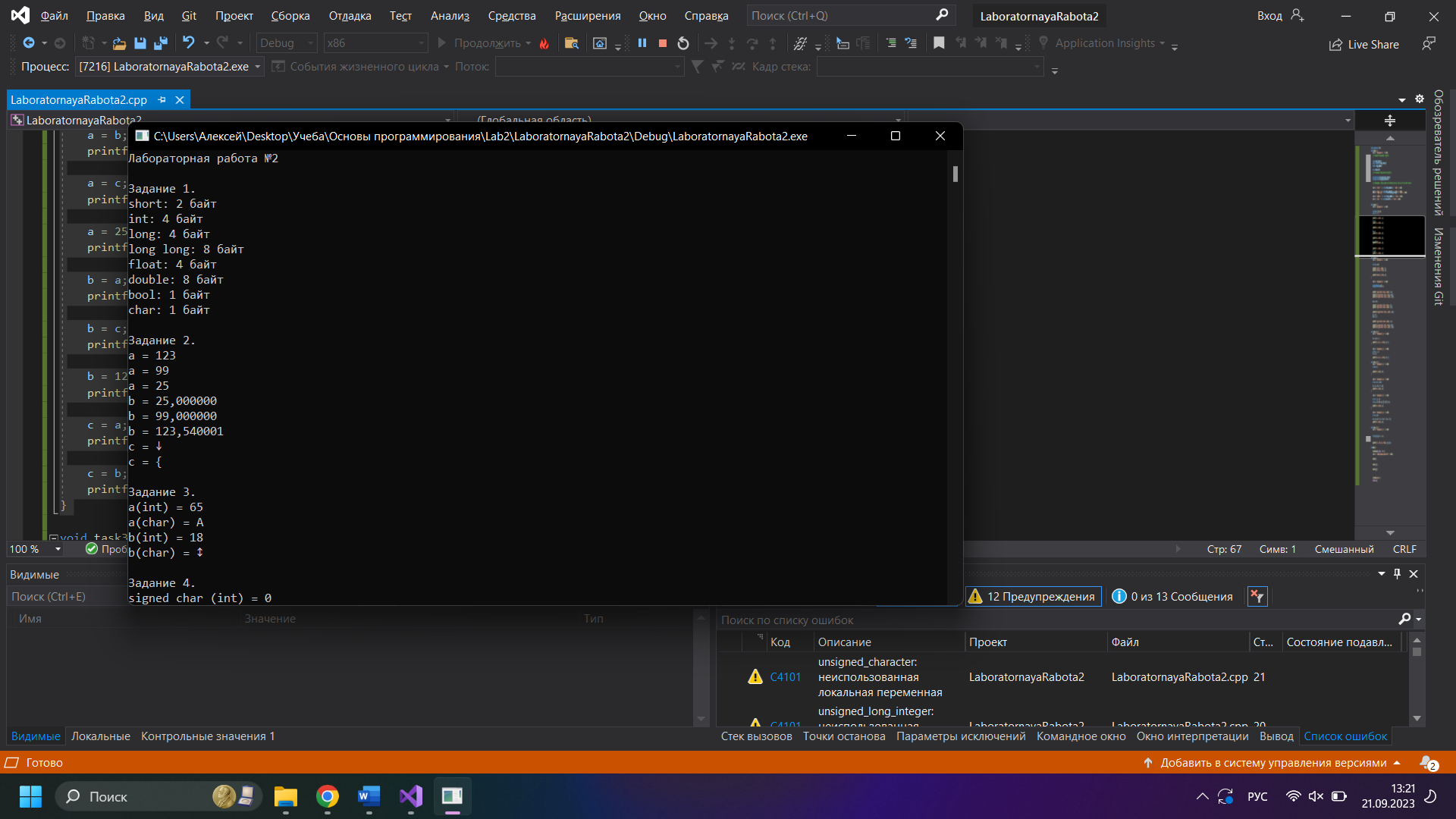


Рисунок 2 – Результат выполнения кода задания 2

При использовании шаблонов форматирования в функции printf происходит автоматическая конвертация аргумента по шаблону:

* %i – к целому числу;
* %f – к числу с плавающей точкой;
* %c – к ASCII символу.

Таким образом можно заметить, что при проведении чисел с плавающей точкой к целочисленному происходит отбрасывание дробной части, а при конвертации символа мы получаем его ASCII код.

При попытке приведения к числу с плавающей точкой можно заметить добавление дробной части к числу.

При выводе как символ изначально отбрасывается дробная часть, если такова имеется, а после этого выводится символ из ASCII таблицы.

**Задание 3.**

Переполнение типов.

* Объявить две символьные переменные, проинициализировать их значениями 65 и 274, вывести значения каждой переменной со спецификаторами для целого и символьного типов.

Листинг приложения:

void task3() {

cout << "\nЗадание 3." << endl;

char a = 65;

char b = 274;

printf("a(int) = %i\n", a);

printf("a(char) = %c\n", a);

printf("b(int) = %i\n", b);

printf("b(char) = %c\n", b);

}

Результат выполнения кода:

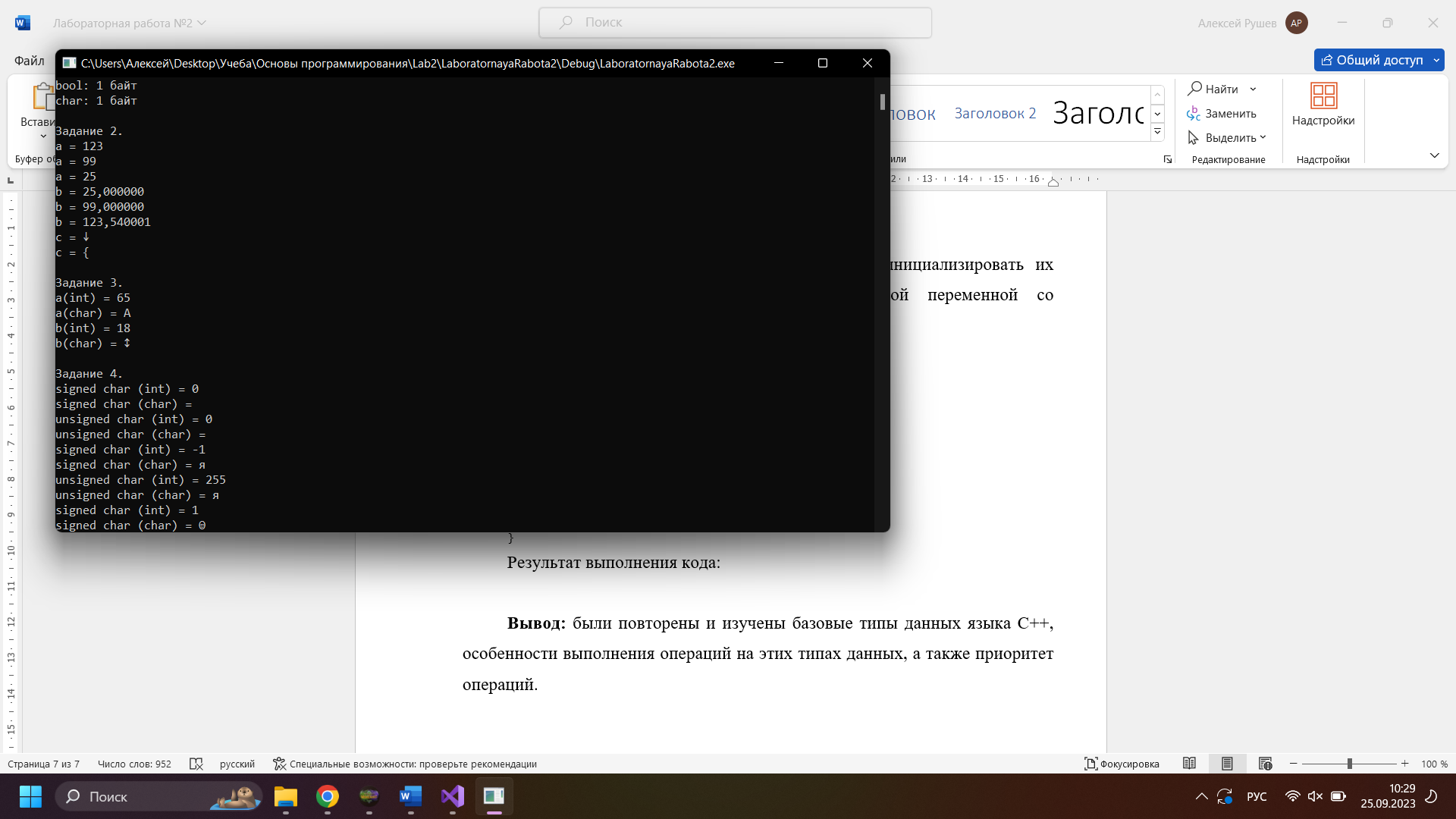


Рисунок 3 – Результат выполнения кода задания 3.

**Задание 4.**

* Объявить две переменные следующих типов: signed char и unsigned char, проинициализировать их значением 0, вывести значения переменных на экран со спецификаторами для целого знакового и символьного типов. Задать значение объявленных выше переменных равным 255, вывести на экран с указанными выше спецификаторами, увеличить значение каждой переменной на два, вывести на экран изменённые значения со спецификаторами для целого знакового и символьного типов.

Листинг приложения:

void task4() {

cout << "\nЗадание 4." << endl;

signed char sc = 0;

unsigned char usc = 0;

printf("signed char (int) = %i\n", sc);

printf("signed char (char) = %c\n", sc);

printf("unsigned char (int) = %i\n", usc);

printf("unsigned char (char) = %c\n", usc);

sc = 255;

usc = 255;

printf("signed char (int) = %i\n", sc);

printf("signed char (char) = %c\n", sc);

printf("unsigned char (int) = %i\n", usc);

printf("unsigned char (char) = %c\n", usc);

sc += 2;

usc += 2;

printf("signed char (int) = %i\n", sc);

printf("signed char (char) = %c\n", sc);

printf("unsigned char (int) = %i\n", usc);

printf("unsigned char (char) = %c\n", usc);

}

Результат выполнения кода:

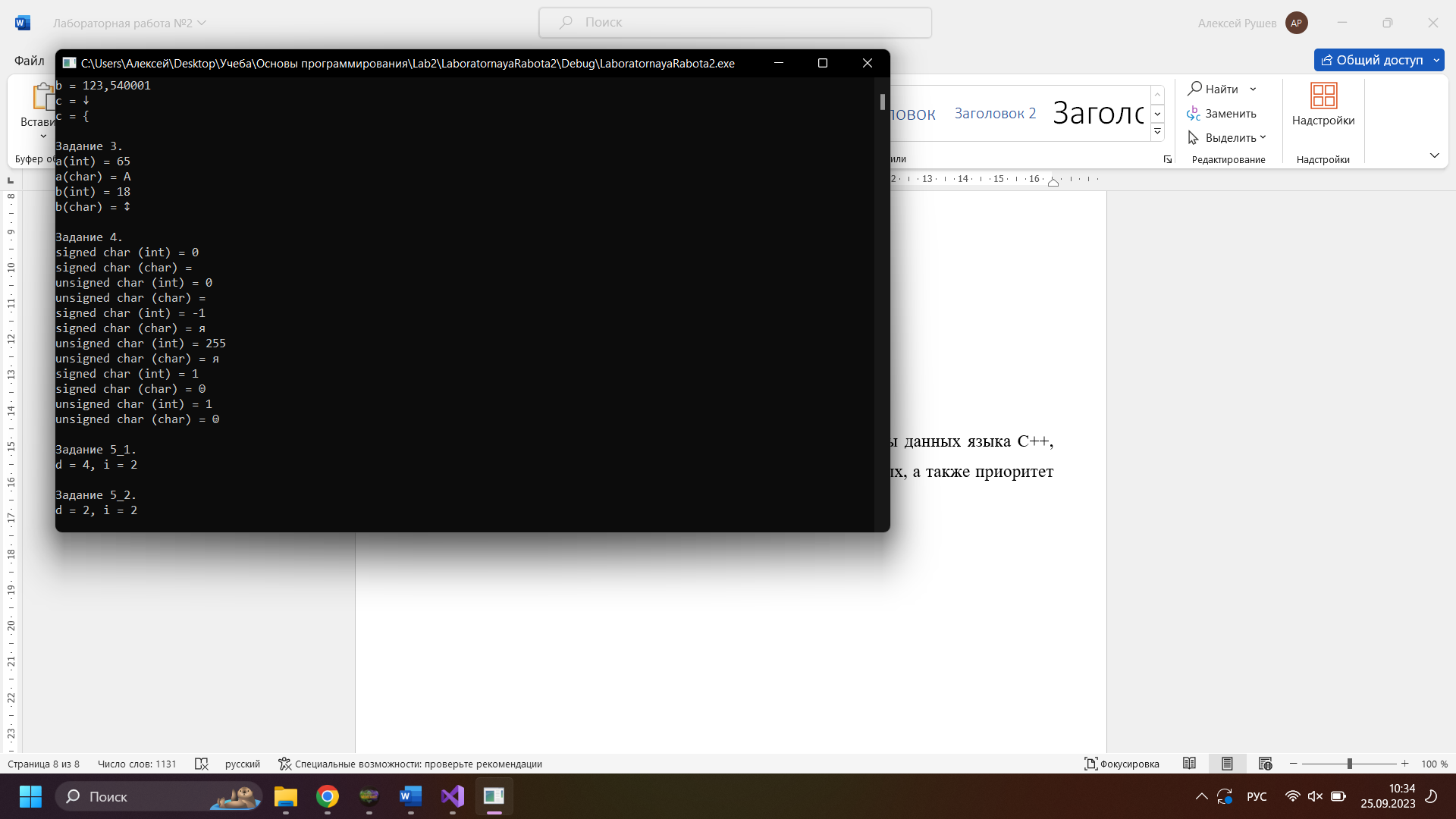


Рисунок 4 – Результат выполнения кода задания 4.

В результате выполнения программы выше, мы выяснили, что при переполни беззнаковых переменных типа символ отсчёт начинается с нуля, а при переполнении знаковых отсчёт начнётся с -128.

**Задание 5.**

Приоритет операций.

Листинг приложения.

void task5\_1() {

cout << "\nЗадание 5\_1." << endl;

int d, i = 1;

d = (++i) + 2;

printf("d = %i, i = %i\n", d, i);

}

Результат выполнения кода.

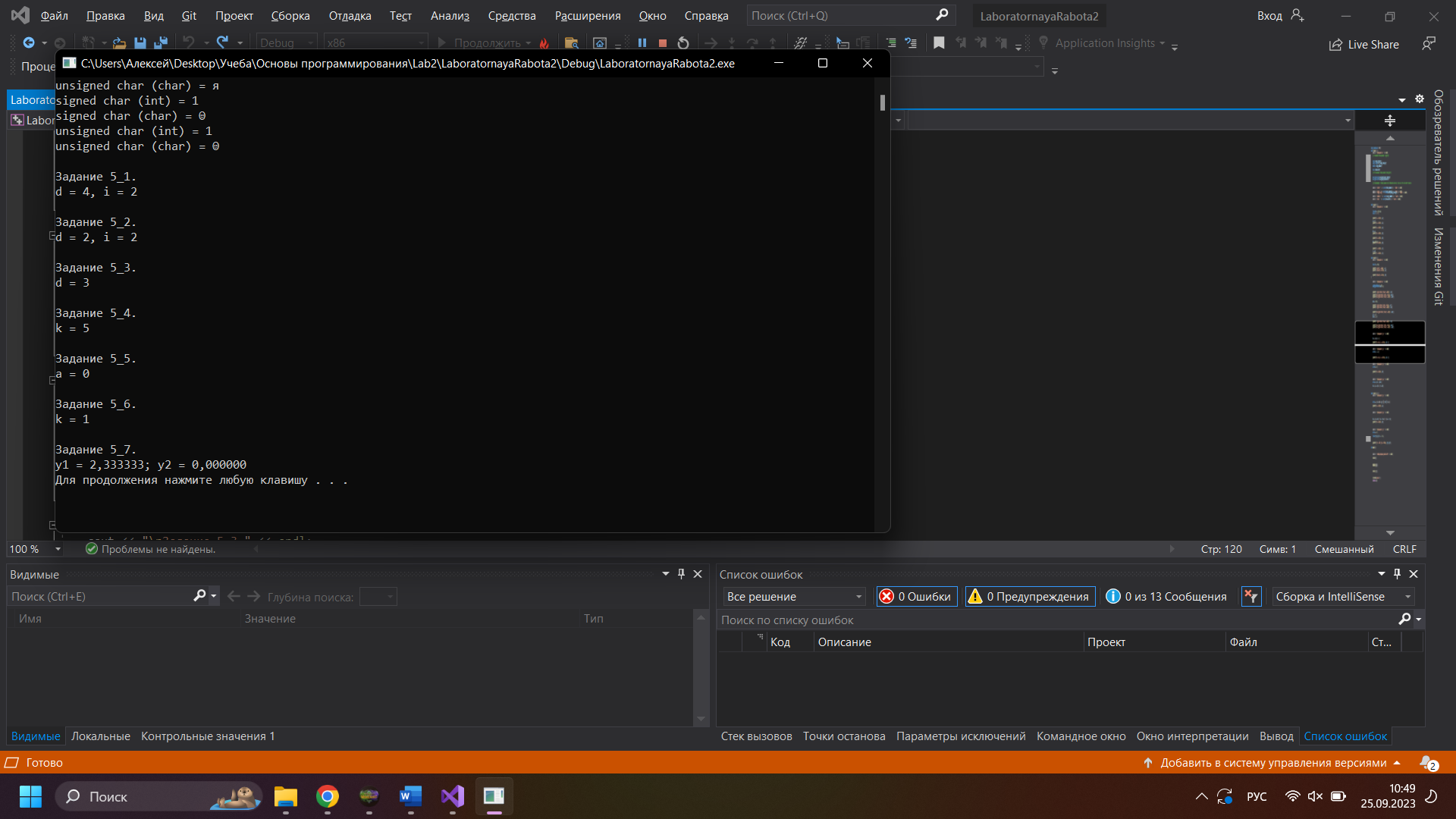


Рисунок 5 – Результат выполнения кода

Последовательность действий:

1. Объявление переменных d и i;
2. Присваивание единицы переменной i;
3. Инкремент и возвращение нового значения переменной i;
4. Запись результата сложения в переменную d.

Листинг приложения.

void task5\_2() {

cout << "\nЗадание 5\_2." << endl;

int d, i = 1;

d = ++i;

printf("d = %i, i = %i\n", d, i);

}

Результат выполнения кода.

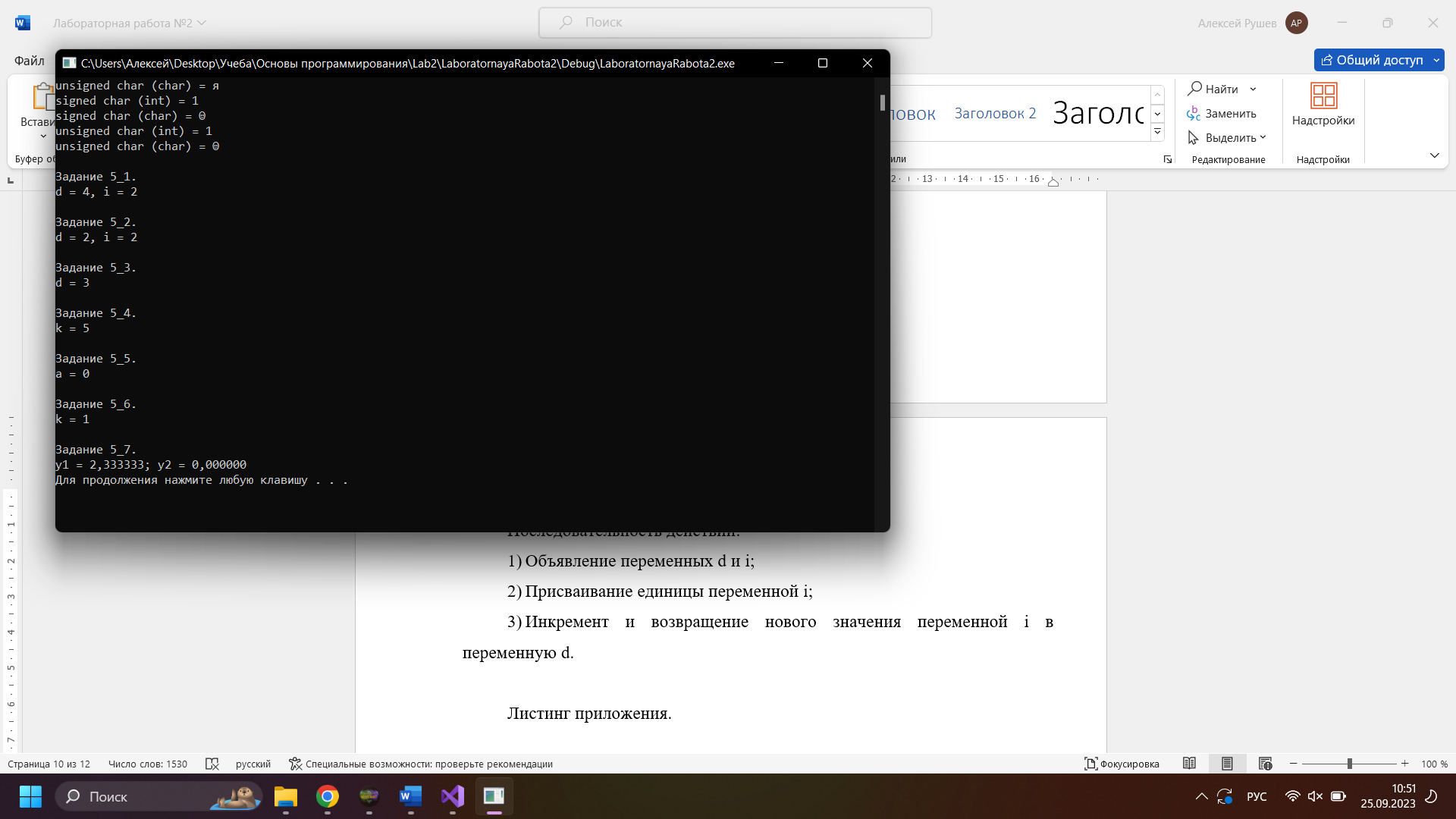


Рисунок 6 – Результат выполнения кода

Последовательность действий:

1. Объявление переменных d и i;
2. Присваивание единицы переменной i;
3. Инкремент и возвращение нового значения переменной i в переменную d.

Листинг приложения.

void task5\_3() {

cout << "\nЗадание 5\_3." << endl;

int d = 1;

d += d++;

printf("d = %i\n", d);

}

Результат выполнения кода.

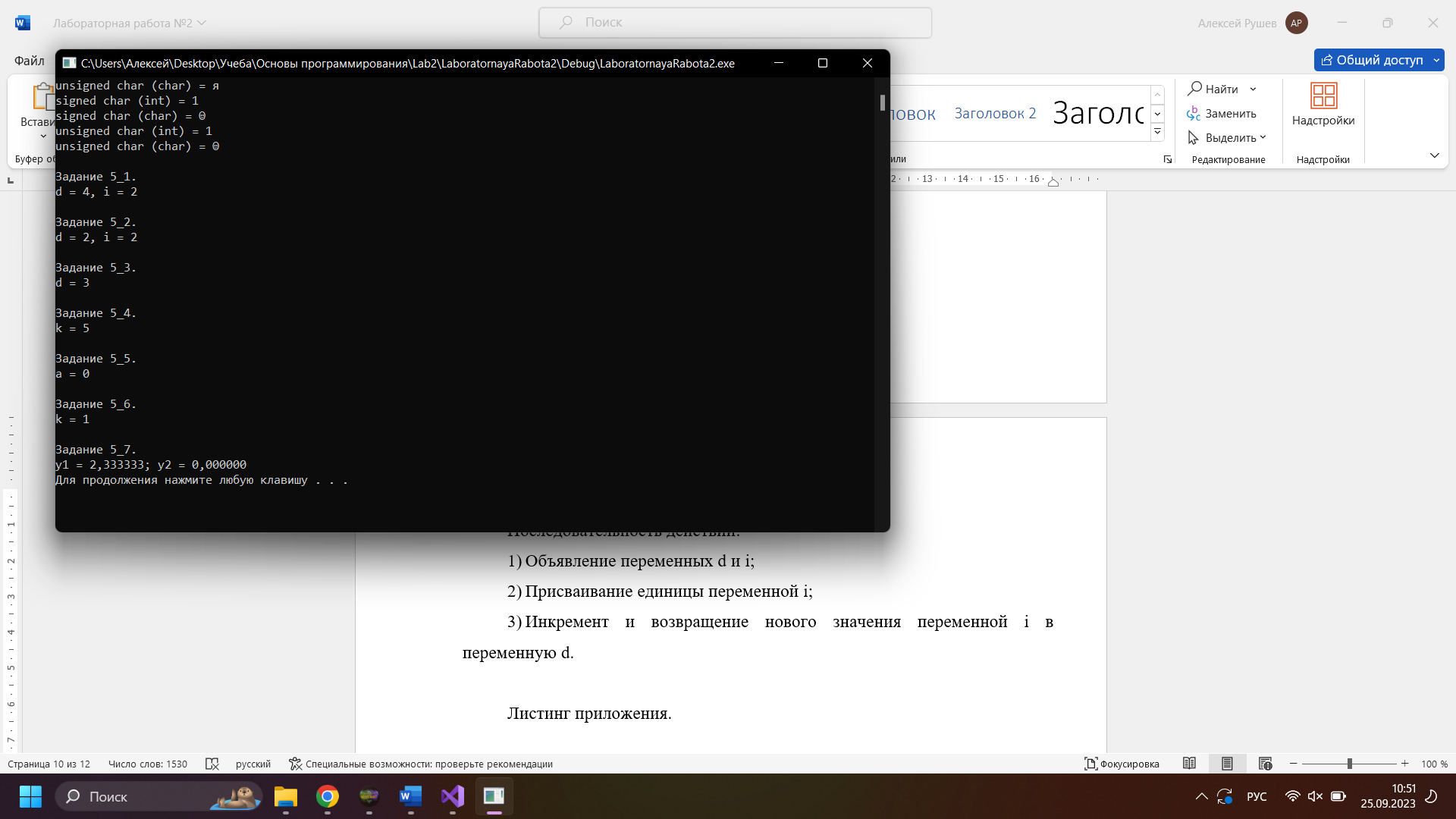


Рисунок 7 – Результат выполнения кода

Последовательность действий:

1. Объявление переменной d со значением 1;
2. Инкремент и возвращение старого значения переменной d, сумма с новым значением d.

Листинг приложения.

void task5\_4() {

cout << "\nЗадание 5\_4." << endl;

int a, b, c, d, k;

b = 2; d = 3;

k = (a = b) + (c = d);

printf("k = %i\n", k);

}

Результат выполнения кода.

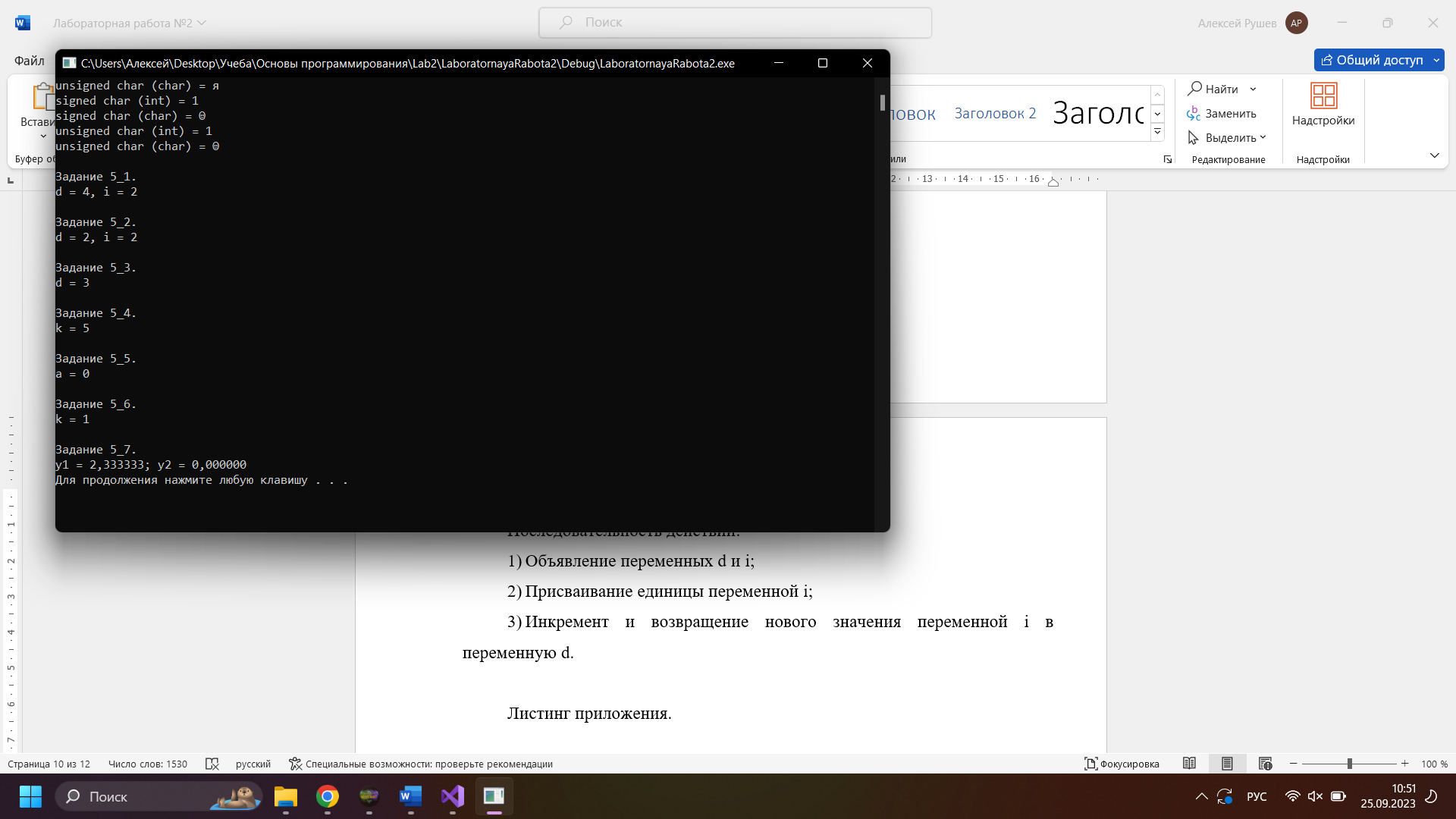


Рисунок 8 – Результат выполнения кода

Последовательность действий:

1. Объявление переменной a, b, c, d, k;
2. Присвоение значений переменной b и переменной d;
3. Присвоение значения переменной a из переменной b;
4. Присвоение значения переменной c из переменной d;
5. Сложение.

Листинг приложения.

void task5\_5() {

cout << "\nЗадание 5\_5." << endl;

int i, l, j, k;

i = l = j = k = 0;

int a = (i++ && ++j) || (k || l++);

printf("a = %i\n", a);

}

Результат выполнения кода.

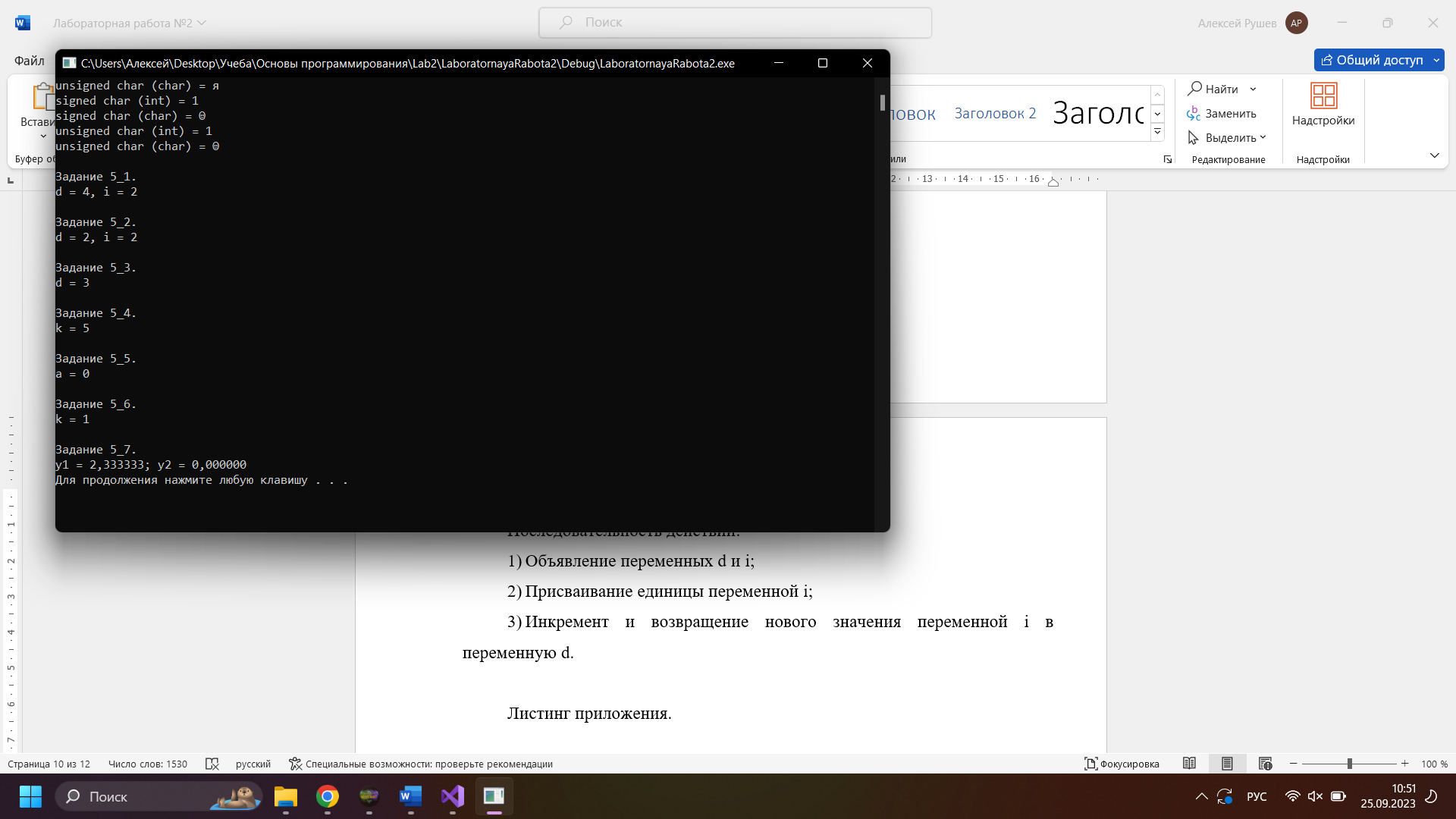


Рисунок 9 – Результат выполнения кода

Последовательность действий:

1. Объявление переменных I, l, j, k;
2. Присваивание переменным I, l, j, k значение равное нулю;
3. Инкремент переменной i;
4. Инкремент переменной j;
5. Логическая операция И над переменными i и j;
6. Инкремент переменной l;
7. Логическое ИЛИ над переменными k и l;
8. Логическое ИЛИ над результатом двух скобок.

Листинг приложения.

void task5\_6() {

cout << "\nЗадание 5\_6." << endl;

int a, b, k;

a = 2; b = 1;

k = (a != b) ? (a - b++) : (++a - b);

printf("k = %i\n", k);

}

Результат выполнения кода.

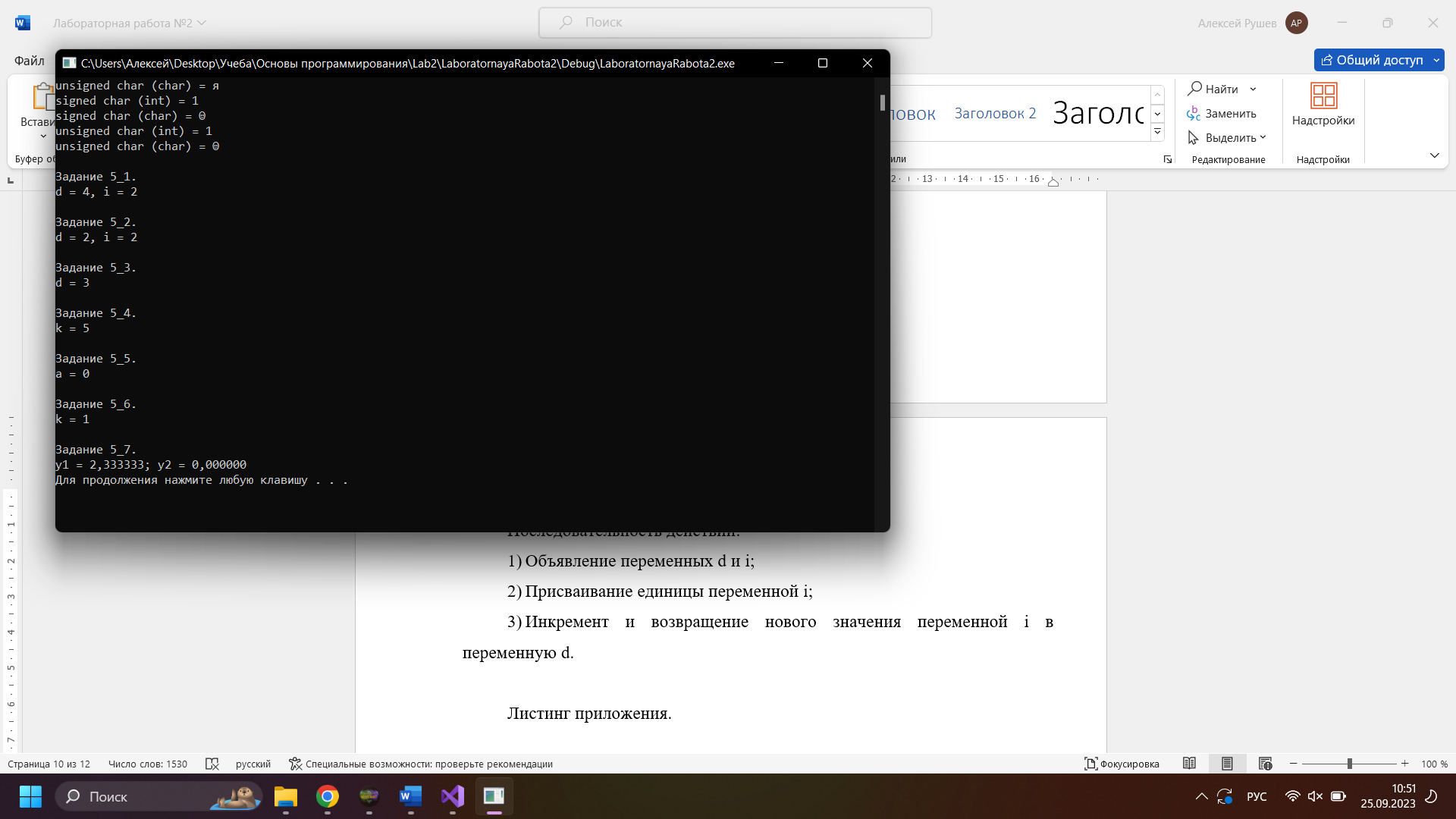


Рисунок 10 – Результат выполнения кода

Последовательность действий:

1. Объявление переменных a, b, k;
2. Присваивание двойки переменной a;
3. Присваивание единицы переменной b;
4. Логическое не равно над переменными a и b;
5. Инкремент переменной b;
6. Вычитание переменной b из a;
7. Запись в переменную k.

Листинг приложения.

void task5\_7() {

cout << "\nЗадание 5\_7." << endl;

int a = 2;

int b = 3;

float y1, y2, c = 3.5;

y1 = c \* a / b;

y2 = c \* (a / b);

printf("y1 = %f; y2 = %f\n", y1, y2);

}

Результат выполнения кода.

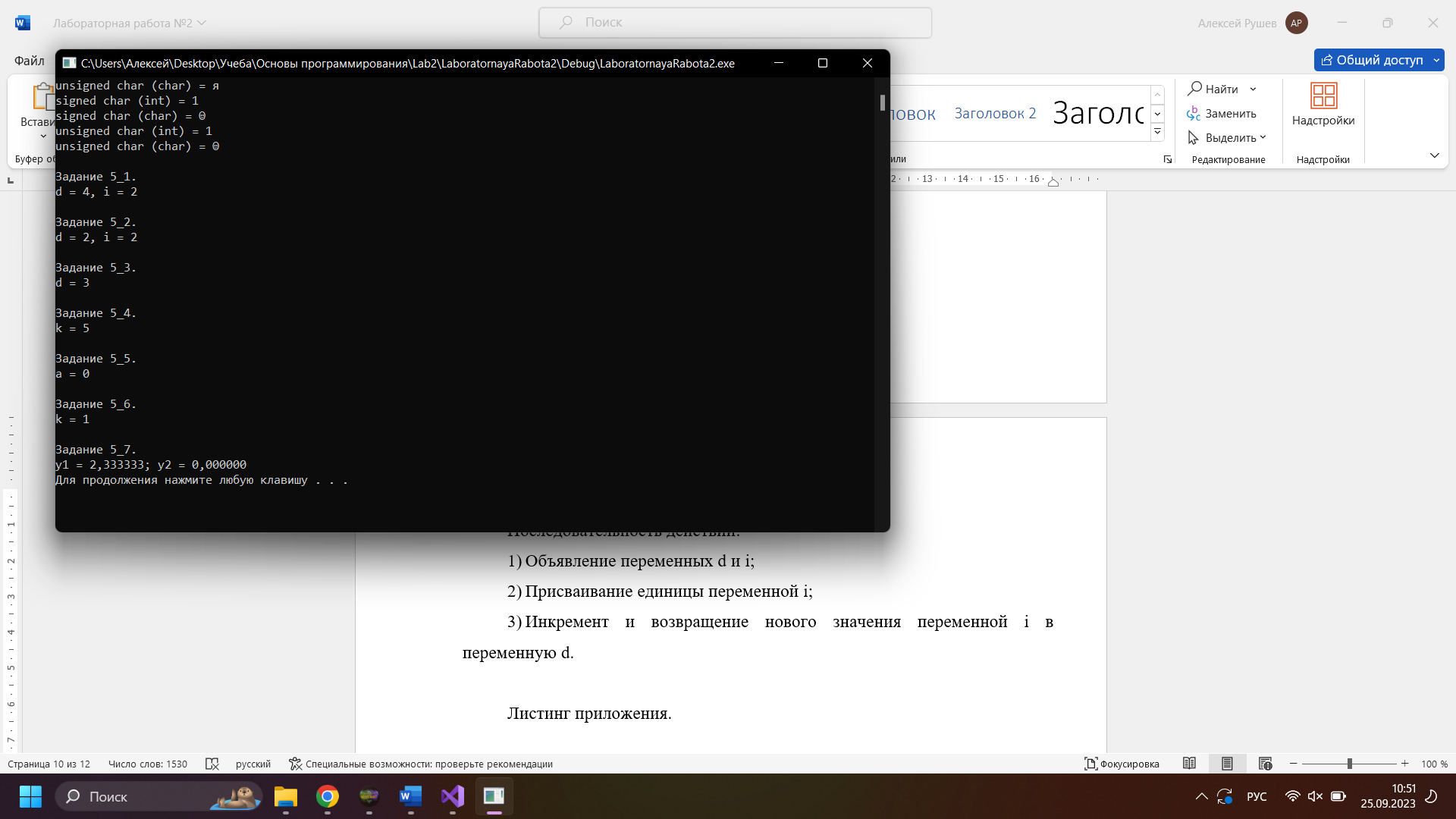


Рисунок 11 – Результат выполнения кода

Последовательность действий:

1. Объявление переменной a со значением 2;
2. Объявление переменной b со значением 3;
3. Объявление переменных y1, y2, c со значением 3.5;
4. Умножение переменной c на a;
5. Деление результата умножения на переменную b;
6. Запись в переменную y1;
7. Деление переменной a на b;
8. Умножение результата деления на переменную c;
9. Запись в переменную y2.

Листинг приложения.

int main()

{

setlocale(LC\_ALL, "rus");

cout << "Лабораторная работа №2" << endl;

task1();

task1\_2();

task2();

task3();

task4();

task5\_1();

task5\_2();

task5\_3();

task5\_4();

task5\_5();

task5\_6();

task5\_7();

system("pause");

return 0;

}

**Вывод:** были повторены и изучены базовые типы данных языка C++, особенности выполнения операций на этих типах данных, а также приоритет операций.