**ЛАБОРАТОРНАЯ РАБОТА №1**

Курс «Объектно-ориентированное программирование»

**Тема:** Синтаксис языка C++. Работа с памятью в С++.

**Цель:** Научиться программировать на языке С++ базовые операции с данными разных типов, разветвляющиеся и циклические алгоритмы, научиться работать с динамической памятью в С++, осуществлять консольный ввод-вывод данных.

**Ход работы**

**Вариант 5**

1. Написать программу обработки одномерного массива с n элементами, в соответствии с вариантом (приложение А). Необходимые действия должны производиться в функции **processArray()**, возвращающей определенное значение и формирующей еще один (выходной) массив данных (см. свой вариант). Память под массивы выделять статически (объявлять массивы локально в функции **main**) и для доступа к элементам использовать индексы. Ввод-вывод данных организовать средствами cstdio.

Объявить массив из n=16 целых чисел, проинициализировать единицами. Функция processArray() должна заполнить элементы массива с четными индексами степенями двойки (1, 2, 4, 8, 16, …), с нечетными индексами – степенями тройки (3, 9, 27, …). Также подсчитать и вернуть count – количество двузначных чисел в массиве и сформировать выходной массив, содержащий только такие числа. Вывести на экран результирующие массивы.

Вход: [ 3 2 9 4 27 8 81 16 243 32 729 64 2187 128 6561 256 ]

Выход: [ 27 81 16 32 64 ]

#include <iostream>

int processArray(const int arrLength, int count, int inputArr[], int outputArr[]) {

int powerOf2 = 1;

int powerOf3 = 1;

for (int i = 0; i < arrLength; i += 2) {

powerOf2 \*= 2;

inputArr[i] = powerOf2;

}

for (int i = 1; i < arrLength; i += 2) {

powerOf3 \*= 3;

inputArr[i] = powerOf3;

}

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

int digitsAmount = 0;

int inputArrItemCopy = inputArr[i];

while (inputArrItemCopy != 0) {

inputArrItemCopy = inputArrItemCopy / 10;

digitsAmount++;

}

if (digitsAmount == 2) {

outputArr[count] = inputArr[i];

count++;

}

}

return outputArr, count;

}

void printArr(int arr[], int length) {

// вывожу первое значение массива вне цикла, чтобы не было лишнего пробела в выводе

std::cout << arr[0];

for (int i = 1; i < length; i++) {

std::cout << " " << arr[i];

}

}

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

std::cout << "ЛР1. Задание 1\n";

const int arrLength = 16; // длина массивов задаётся в одном месте

int inputArr[arrLength] = { 1 }; // создаём массив и заполняем его единицами

int outputArr[arrLength]; // массив, куда запишутся только двухзначние числа

int count = 0; // количество двухзначных чисел

// записываем в массив outputArr и в count результат работы processArray

outputArr, count = processArray(arrLength, count, inputArr, outputArr);

std::cout << "\nВход: [";

printArr(inputArr, arrLength);

std::cout << "]";

std::cout << "\nВыход: [";

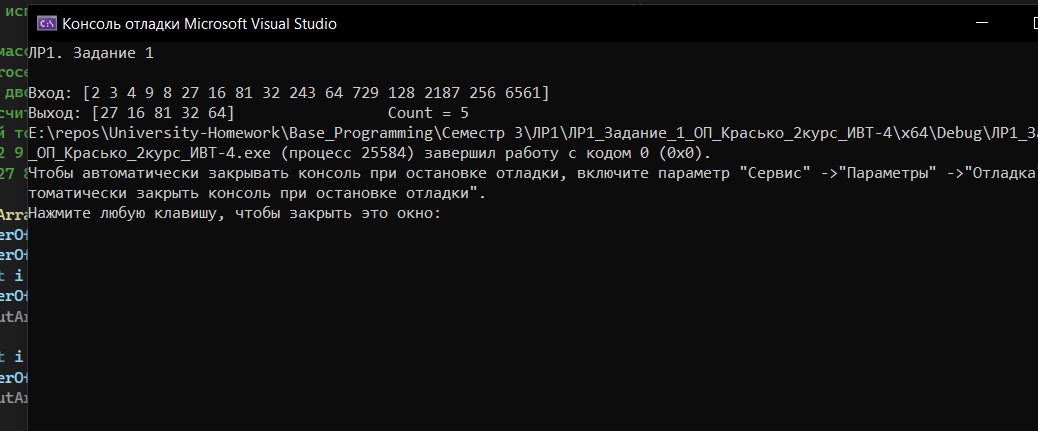
printArr(outputArr, count);

std::cout << "]";

std::cout << "\t\t\tCount = " << count;

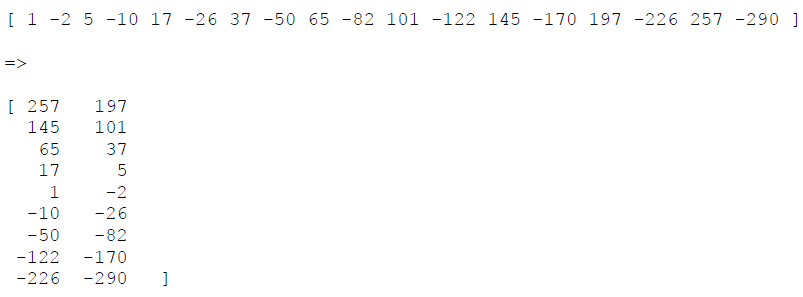
//std::cout << arr[1];

}



1. Написать программу, которая преобразует одномерный массив (1D) в двумерный (2D) (или наоборот), в соответствии с вариантом. Необходимо оформить в отдельных функциях код следующих действий: 1) инициализация массива; 2) вывод массива; 3) преобразование массива (создание нового массива с другой структурой). Память под массивы выделять динамически и для доступа к элементам использовать указатели. Ввод-вывод данных организовать средствами iostream и iomanip.

Преобразование: 1D → 2D. Одномерный массив из 18 целых чисел необходимо отсортировать в порядке убывания и разложить по двумерной сетке 9х2 слева направо и сверху вниз. Инициализация: заполнить массив числами x[i] = i2+1 и все элементы с четными индексами домножить на -1. Вывод на экран: на каждый элемент массива отвести 5 позиций.



// ЛР1\_Задание\_2\_ОП\_Красько\_2курс\_ИВТ-4.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#include <iostream>

#include <iomanip>

using namespace std;

// на вход: длина массива одномерного

// на выход: указатель на одномерный массив

int\* generateArr(int length) {

int\* x = new int[length];

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

x[i] = i \* i + 1;

if (i % 2 == 1) { // В задании написано, что на -1 нужно домножать эл. с чётным индексом

// но в примере умножали вторые элементы по порядку, т.е. с началом отчёта от 1,

// тут я сделал как в примере

x[i] \*= -1;

}

}

return x;

}

//на вход: длина массива, указатель на одномерный массив

//итог: выведенный в консоле массив со скобками квадратными

void print5SymbolArr(int length, int\* arr) {

cout << "[ ";

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

cout << std::setw(5) << arr[i] << " "; //выеделяем 5 символов для каждого элемента массива

}

cout << "]";

}

//на вход: количество столбцов в массиве (т.е. количество массивов в массиве), количество строк в массиве

// (т.е. число элементов в каждом из саб массивов), указатель на двумерный массив

//итог: выведенный в консоле массив в виде матрицы со скобками квадратными

void print5Symbol2DArr(int columnsAmount, int rowsAmount, int\*\* arr) {

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

for (int j = 0; j < rowsAmount; j++) {

if (i == 0 && j == 0) { // для того, чтобы всё выводилось красиво, делаем [ частью 5 символов для вывода первых 5 элементов массива

cout << "[" << std::setw(4) << arr[i][j] << "\t";

}

else {

cout << std::setw(5) << arr[i][j] << "\t";

}

}

if (i != (columnsAmount - 1)) { // для того, чтобы ] была сразу после последнего элемента, а не на новой строке

cout << "\n";

}

}

cout << "]";

}

// на вход: длина массива, указатель на одномерный массив

// итог: отсортированный массив по тому же адрессу

void bubbleSort(int length, int\* arr) {

int temp;

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

for (int j = 0; j < (length - 1 - i); j++) { // берём от элемента 0, до length - i т.к. после каждого внешнего цикла последнее число уже будет минимальным

// отнимаем 1 т.к. мы сравниваем текущий элемент со следующим

if (arr[j] < arr[j + 1]) {

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

// на вход: указатель на одномерный массив, длина одномерного массива, количество желаемый строк, желаемое количество столбцов

// на выход: двумерный массив с суб массивами нужного размера

int\*\* splitArrIntoSmallerOnes(int\* originalArr, int originalLength, int rowsAmount, int columnsAmount) {

// указатель на массив указателей

int\*\* outputArr = new int\*[columnsAmount];

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

// создаём массивы чисел во внутренних массивах [[rowsAmount чисел], [rowsAmount чисел], ..., columnsAmount]

outputArr[i] = new int[rowsAmount];

}

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

for (int j = 0; j < rowsAmount; j++) {

int index = ((i) \* rowsAmount + j); // т.е. для получения инф. о том, с каким элементом начального массива по счёту мы работаем, умножаем индекс столбца на индекс строки и добавляем текущий id строки

if (index > (originalLength - 1)) { // отнимаем 1, чтобы от номера по порядку получить id

// тут если уже все элементы начального массива записаны, остальное заполняется нулями

outputArr[i][j] = 0;

}

else{

outputArr[i][j] = originalArr[index];

}

}

}

return outputArr;

}

int main()

{

setlocale(LC\_ALL, "Russian");

int\* arr1D{ nullptr };

const int arrLength = 18;

arr1D = generateArr(arrLength);

bubbleSort(arrLength, arr1D);

print5SymbolArr(arrLength, arr1D);

cout << "\n=>";

int columnsAmount, rowsAmount;

cout << "\nВведите размер групп, на которые нужно разбить массив: ";

while (true) { // проверка правильности ввода

cin >> rowsAmount;

if (cin.fail() || rowsAmount <= 0) {

cin.clear(); // Очищаем флаг ошибки ввода

cin.ignore(numeric\_limits<streamsize>::max(), '\n'); // Игнорируем оставшийся ввод

cout << "Ошибка ввода. Введите положительное число: ";

}

else {

break;

}

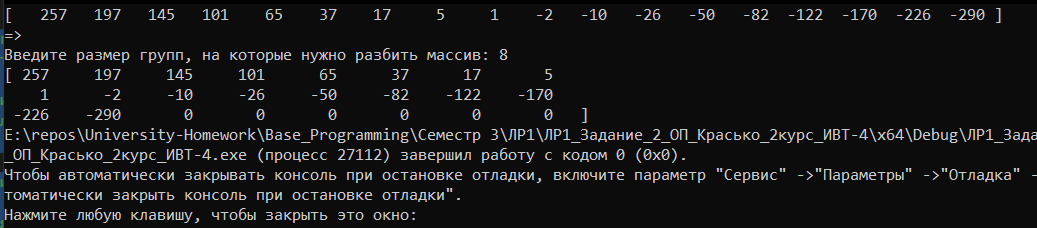
}

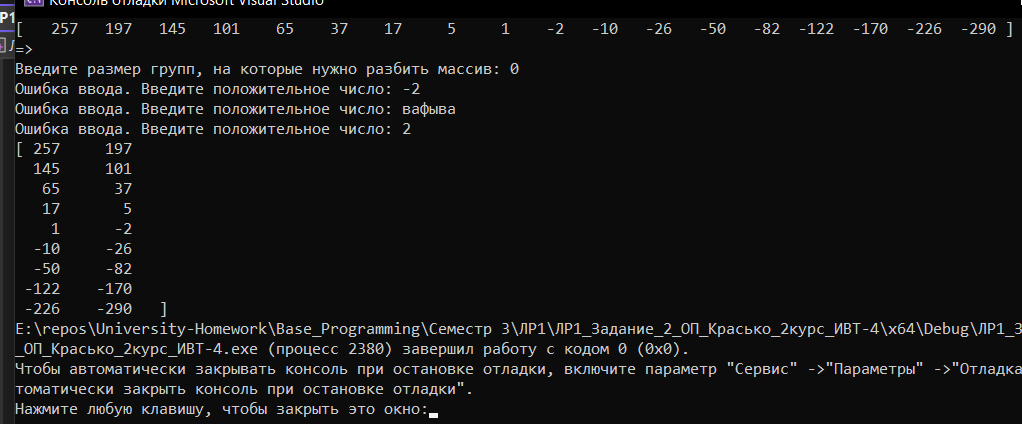
columnsAmount = (arrLength + rowsAmount - 1) / rowsAmount; // округление в большую сторону

int\*\* arr2D = splitArrIntoSmallerOnes(arr1D, arrLength, rowsAmount, columnsAmount);

print5Symbol2DArr(columnsAmount, rowsAmount, arr2D);

}





1. Написать свой аналог стандартной функции обработки строк из файла cstring, в соответствии с вариантом. В функции main на тестовых данных продемонстрировать результат работы как стандартной функции, так и собственной версии. Ввод-вывод данных организовать средствами cstdio.



#define \_CRT\_SECURE\_NO\_WARNINGS // чтобы компилятор не жаловался на strcpy

#include <iostream>

#include <Windows.h> // нужно для ввода пользователем русских символов

using namespace std;

void printString(char\* string) {

for (int i = 0; i <= strlen(string); i++) {

cout << string[i];

}

cout << "\n";

}

void copyString(char\* whereToCopy, char\* whatToCopy) {

for (int i = 0; i <= strlen(whatToCopy); i++) {

whereToCopy[i] = whatToCopy[i];

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

SetConsoleOutputCP(1251); // нужно для ввода пользователем русских символов

SetConsoleCP(1251); // нужно для ввода пользователем русских символов

const int MAX\_LENGTH = 100;

// начальная строка от пользователя

cout << "Введите строку: ";

char\* src = new char[MAX\_LENGTH];

std::cin.getline(src, MAX\_LENGTH);

// заводская функция

char\* dest = new char[strlen(src) + 1]; // добавляем 1 байт, чтобы было место для символа завершения строки \0

strcpy(dest, src);

cout << "Результат от strcpy:\n";

printString(dest);

// самодельная функция

char\* dest2 = new char[strlen(src) + 1];

copyString(dest, src);

cout << "Результат от собственноручной функции:\n";

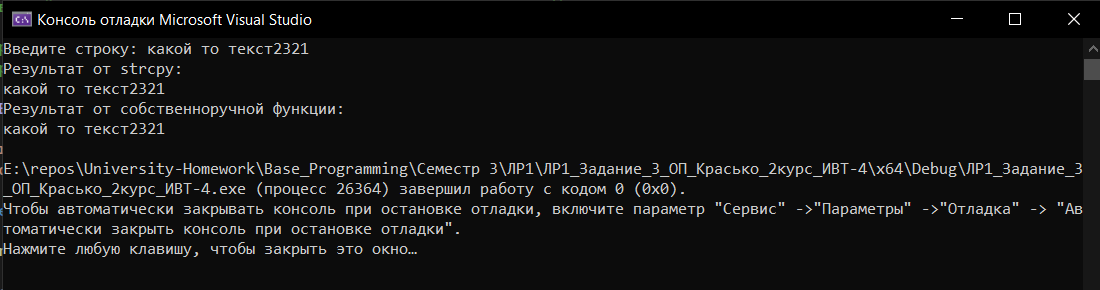
printString(dest);

delete[] src;

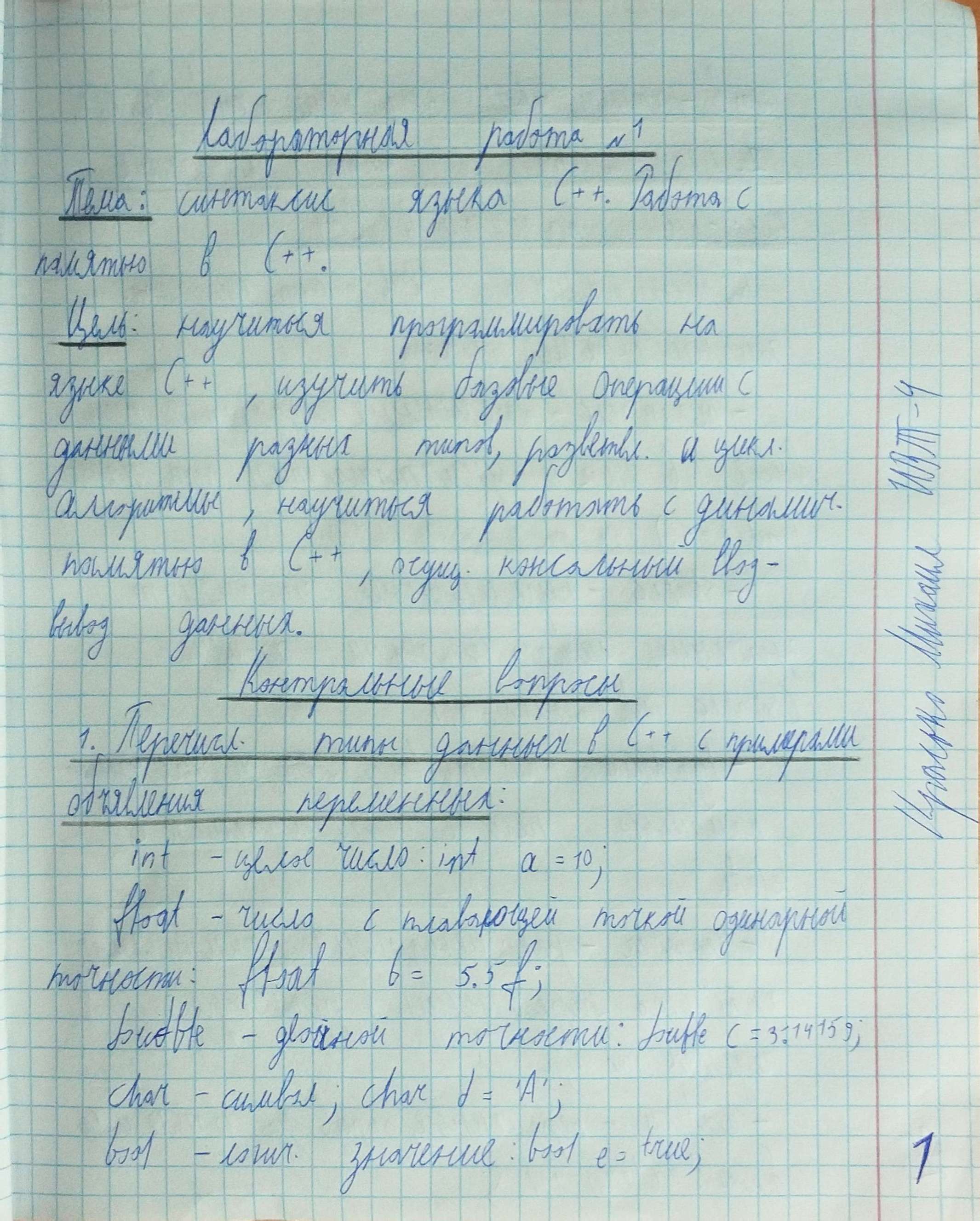
delete[] dest;

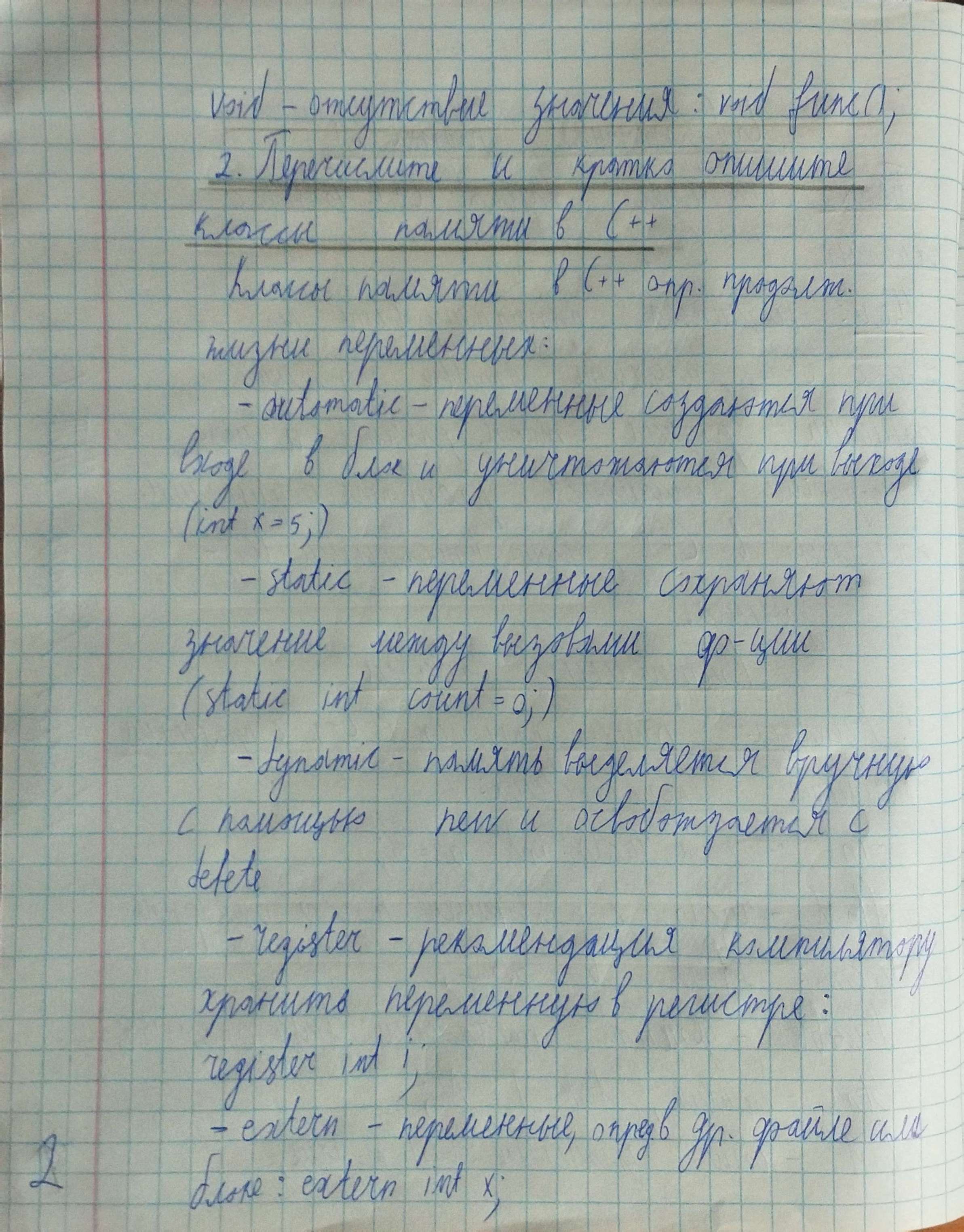
delete[] dest2;

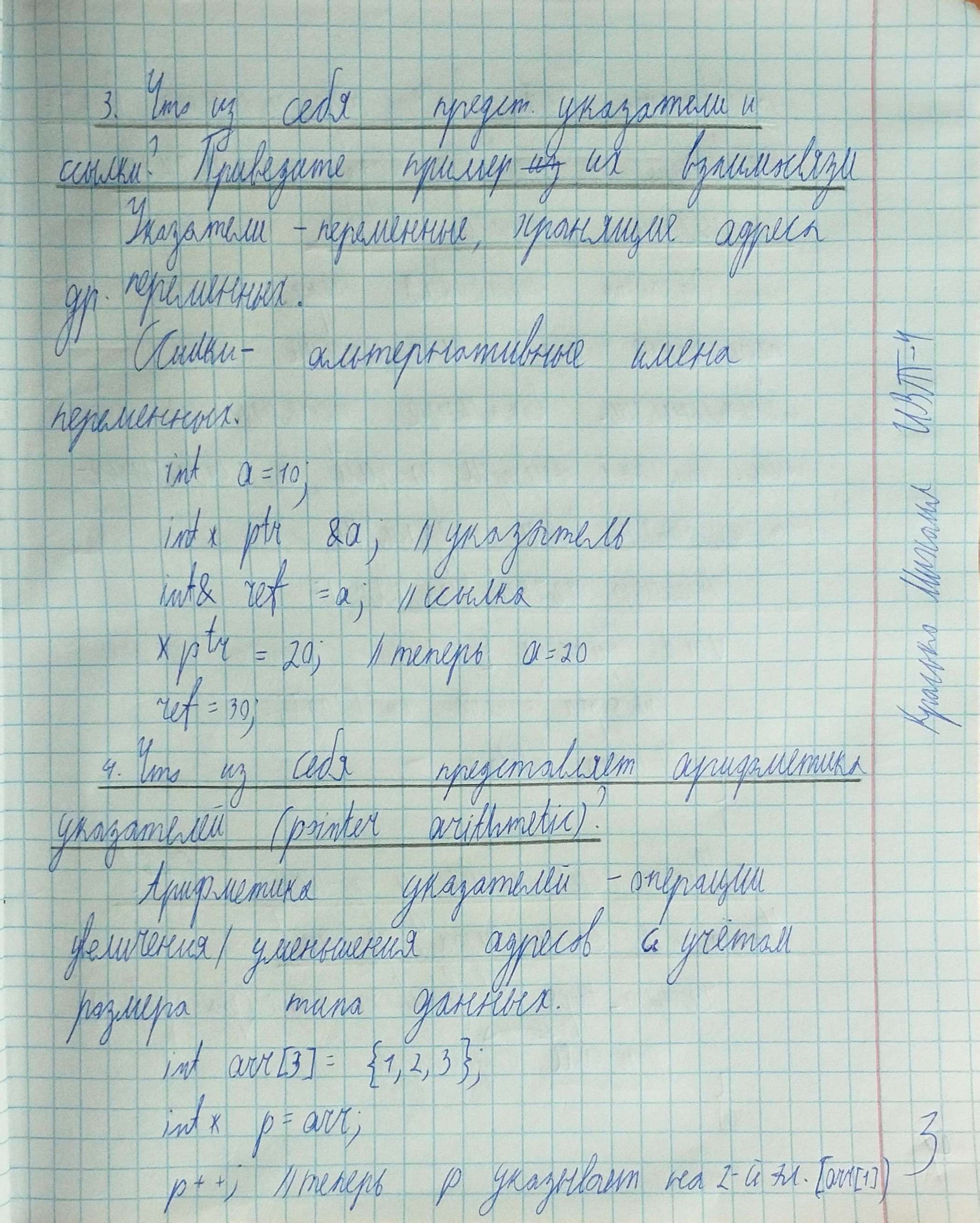
}

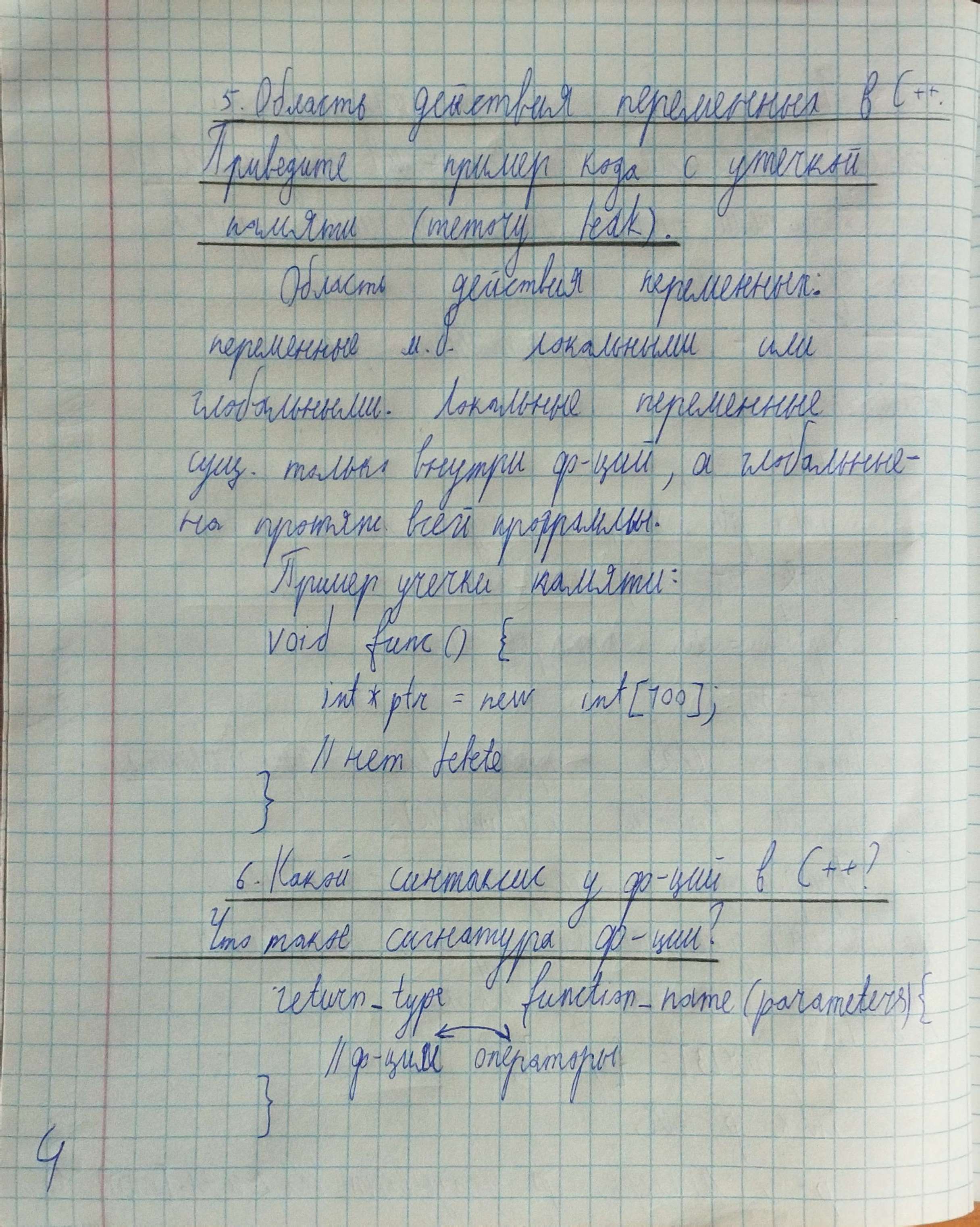


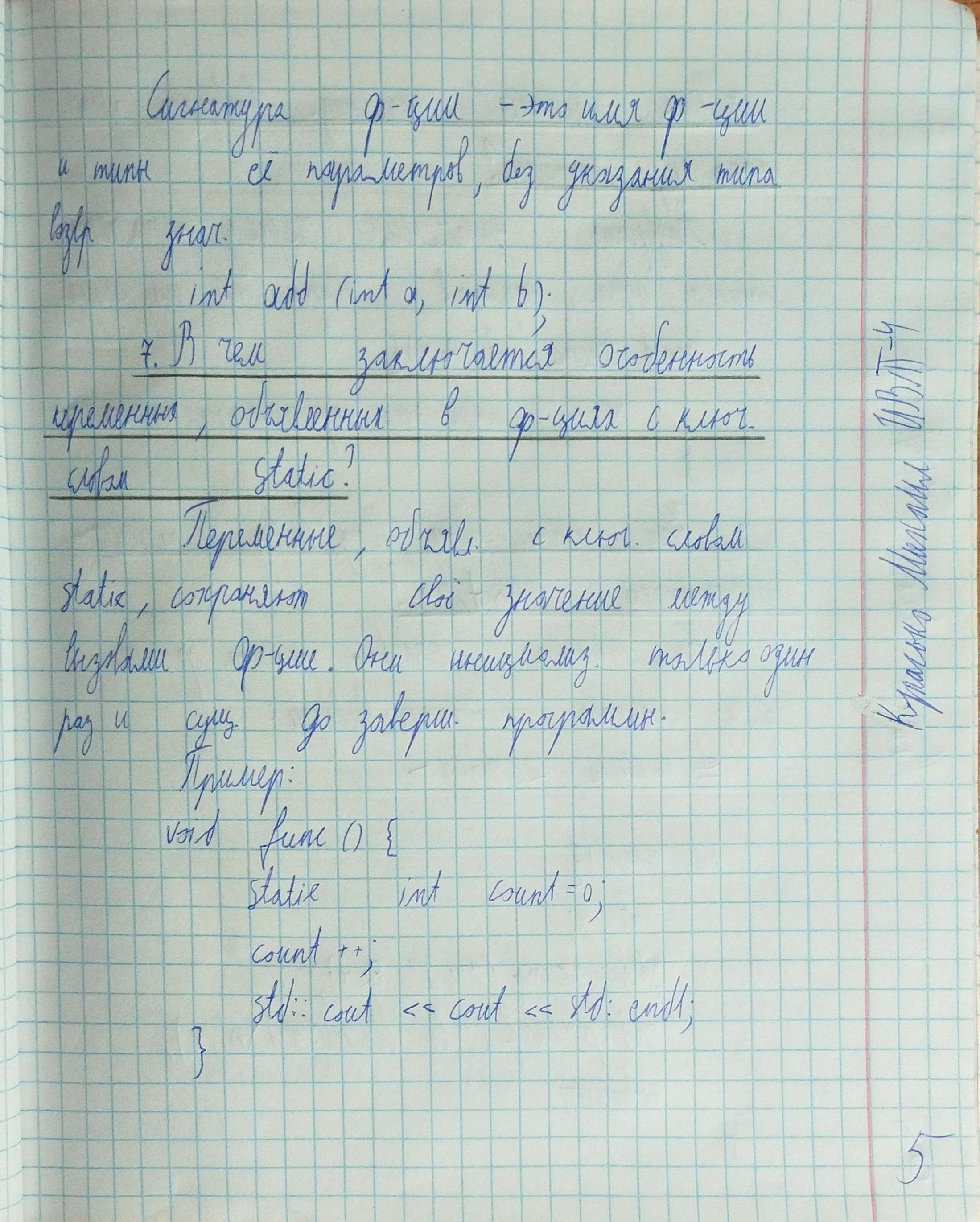
Контрольные вопросы

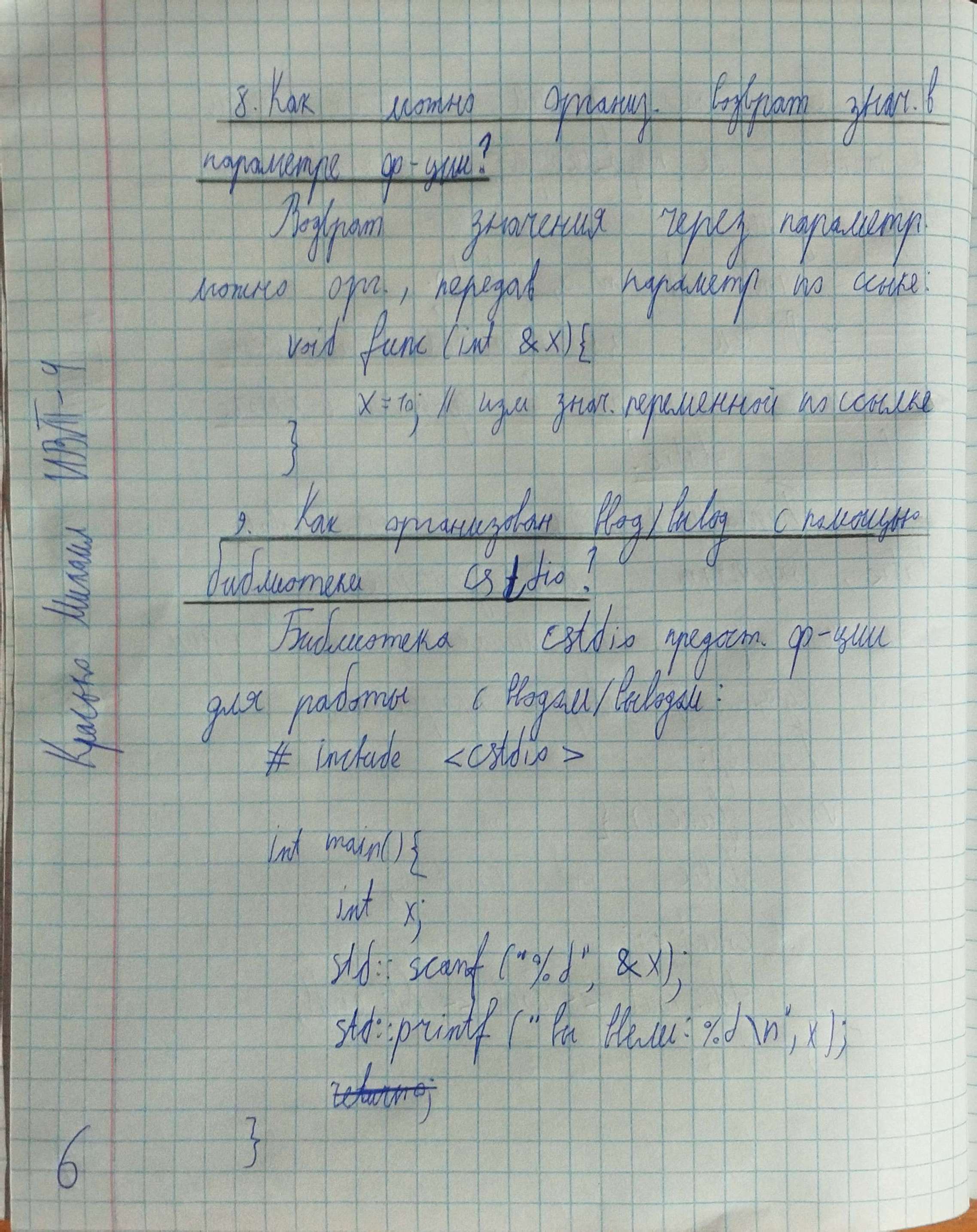


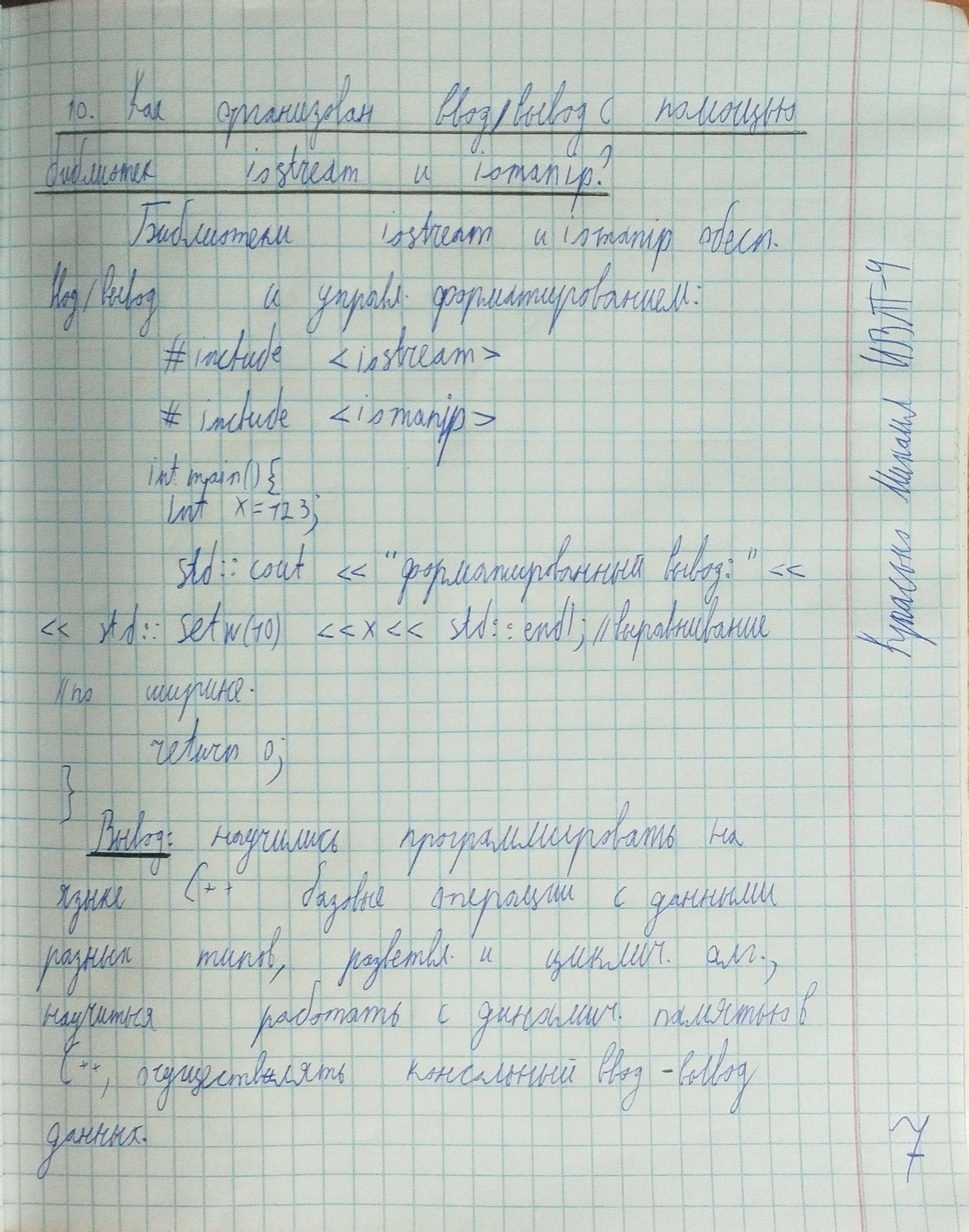












**Вывод:** научились программировать на языке С++ базовые операции с данными разных типов, разветвляющиеся и циклические алгоритмы, научились работать с динамической памятью в С++, осуществлять консольный ввод-вывод данных.