Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Иркутский национальный исследовательский технический

университет»

Институт информационных технологий и анализа данных

**О Т Ч Ё Т**

|  |  |  |
| --- | --- | --- |
| о прохождении | | учебной практики |
|  | | (вид практики: учебная/производственная) |
| технологической (проектно-технологической) практики | | |
| (тип практики: технологическая/научно-исследовательская работа/преддипломная и др.) | | |
|  | | |
| в | ИРНИТУ | |
|  | (наименование профильной организации) | |

Обучающегося Прохоренко С.А., ИСИБ-24-1

****(ФИО, группа, подпись)

Руководитель практики от института ИТиАД

Кононенко Р.В., доцент института ИТиАД

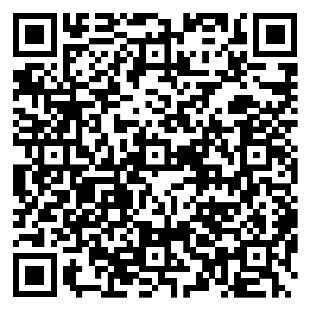
(ФИО, должность, подпись)

Руководитель образовательной программы

Кононенко Р.В., доцент института ИТиАД

<https://irkutsk.hh.ru/resume/a1b28db7ff0efd794c0039ed1f3773356a6e78>

(ФИО, должность, подпись)

**** Оценка по практике

(ФИО, подпись, дата)

Содержание отчета на \_\_\_ стр.

Приложение к отчету на \_\_\_ стр.

<https://irkutsk.superjob.ru/resume/programmist-55733076.html>

Иркутск 2025

**Индивидуальное задание на прохождение**

**учебной практики: технологической (проектно-технологической) практики**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| для | Прохоренко Сергея Андреевича | | | | |
|  | (ФИО обучающегося полностью) | | | | |
| обучающегося | | 1 | курса | группы | ИСИБ-24-1 |

по направлению подготовки Информационные системы и технологии

профиль Интеллектуальные системы обработки информации и управления

Место прохождения практики: ИРНИТУ

Сроки прохождения практики с «16» июня 2025 г. по «29» июня 2025 г.

Цели и задачи прохождения практики:

Содержание практики, вопросы, подлежащие изучению:

Планируемые результаты практики:

Руководитель практики от

института ИТиАД

\_\_\_\_\_\_\_\_\_\_\_/ Кононенко Р.В. /

(подпись)

**Согласовано:**

Руководитель ООП

\_\_\_\_\_\_\_\_\_\_\_/Кононенко Р.В./

(подпись)

«\_\_\_»\_\_\_\_\_\_\_\_\_\_2025 г.

С настоящим индивидуальным заданием и с программой практики ознакомлен, задание принято к исполнению

\_\_\_\_\_\_\_\_\_\_«\_\_\_»\_\_\_\_\_\_\_\_2025 г.

(подпись)

**ДНЕВНИК**

прохождения практики

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| обучающегося | | | | | Прохоренко Сергея Андреевича, ИСИБ-24-1 |
|  | | | | | (фамилия, имя, отчество, группа) |
| курс | | 1 | | | |
| направление | | | | Информатика и вычислительная техника | |
| профиль | | | Интеллектуальные системы обработки | | |
| информации и управления | | | | | |
| в | ИРНИТУ | | | | |
|  | (наименование профильной организации) | | | | |

Иркутск 2025

Руководителем практики от структурного подразделения назначен:

Кононенко Роман Владимирович, доцент института ИТиАД

(ФИО, должность)

**Рабочий график (план) прохождения практической подготовки**

(заполняется обучающимся)

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Период  практики | Содержание выполненных работ | Подпись руководителя практики от структурного подразделения |
| 1 | 16.06.2025 | Решил задачу №2,  Решил задачу №3,  Изучил теоретический материал для задачи №1. |  |
| 2 | 17.06.2025 | Решил задачу №1. |  |
| 3 | 18.06.2025 | Решил задачу №4,  Решил задачу №6,  Решил задачу №7,  Изучил теоретический материал для задачи №5. |  |
| 4 | 19.06.2025 | Решил задачу №5. |  |
| 5 | 20.06.2025 | Решил задачу №8,  Решил задачу №9,  Решил задачу №10. |  |
| 6 | 23.06.2025 | Решил задачу №11. |  |
| 7 |  |  |  |
| 8 |  |  |  |
| 9 |  |  |  |
| 10 |  |  |  |

Таблица 1 Рабочий график

|  |  |
| --- | --- |
| Дата фактического прибытия |  |
| обучающегося в структурное подразделение | 16.06.2025 |
| Дата фактического убытия |  |
| обучающегося из структурного подразделения | 28.06.2025 |

|  |  |
| --- | --- |
| Руководитель образовательной программы | Кононенко Р.В. |
|  | (ФИО, подпись) |
| Директор института | Говорков А.С. |
|  | (ФИО, подпись) |

**Содержание**

**Введение**

**Задача №1**

Незнайка в своей экспедиции на Луну оказался на вершине лунной горы. Спуск вниз опасен, поэтому он взял с собой карту склона горы, где числами обозначено, сколько минут требуется на этот участок маршрута. Спуск происходит сверху вниз на один из соседних участков. Пример наиболее короткого маршрута выделен красным цветом, сумма чисел = 10. Напишите программу, рассчитывающую минимальное время спуска (сумму чисел в пути с вершины до основания).

**Алгоритм программы:**

1.**Ввод данных:** пользователь вводит высоту пирамиды n.

2.**Генерация пирамиды:** создаётся двумерный вектор pyramid размером n строк. Каждая строка i содержит i+1 случайных чисел (от 1 до 100).

3.**Расчёт минимального пути (динамическое программирование)**: создаётся копия пирамиды — dp. Заполнение идёт снизу вверх: для каждого элемента dp[i][j] прибавляем минимум из двух нижних соседей dp[i+1][j] и dp[i+1][j+1]. В dp[0][0] хранится минимальная сумма.

4.**Восстановление пути:** начинаем с вершины pyramid[0][0]. На каждом шаге выбираем соседа снизу с минимальным значением в dp. Записываем выбранные элементы в вектор path.

5.**Вывод результата:** Минимальная сумма - dp[0][0]. Последовательность вершин пути - path.

**Код программы:**

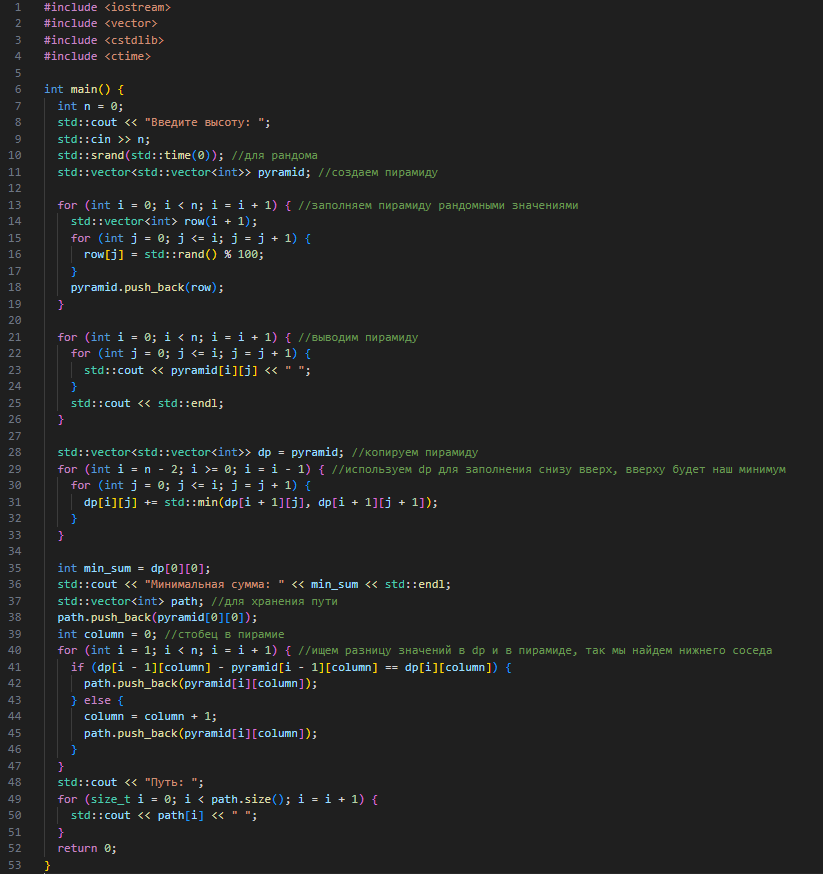
****

Рисунок 1 Код программы задачи 1

**Результаты:**

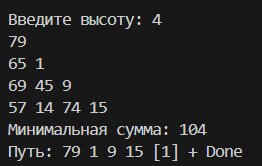
****

Рисунок 2 Результат программы задачи 1

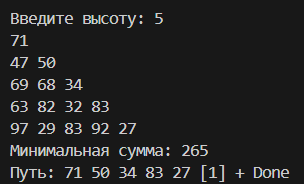
****

Рисунок 3 Результат программы задачи 1

**Задача №2**

После метеоритной атаки компьютерная сеть для управления лунными заводами разбилась на части, нужно объединить её в единое целое. Каждый фрагмент сети представлен в виде ненаправленного графа.Вам известно общее число вершин графа (узлы сети, не более 1000) и набор рёбер (сохранившиеся линии связи, не более 1000).Определите, какое минимальное число линий связи нужно дополнительно построить, чтобы сеть стала единой.

**Алгоритм программы:**

**1. Ввод данных:** читаем количество вершин n и рёбер m. Строим список смежности графа.

**2. Поиск компонент связности:** используем dfs для обхода графа. Каждый вызов dfs помечает все вершины одной компоненты связности. Считаем количество таких компонент (components).

**3. Расчёт результата:** чтобы соединить k компонент, нужно минимум k - 1 рёбер. Выводим components - 1.

**Код программы:**

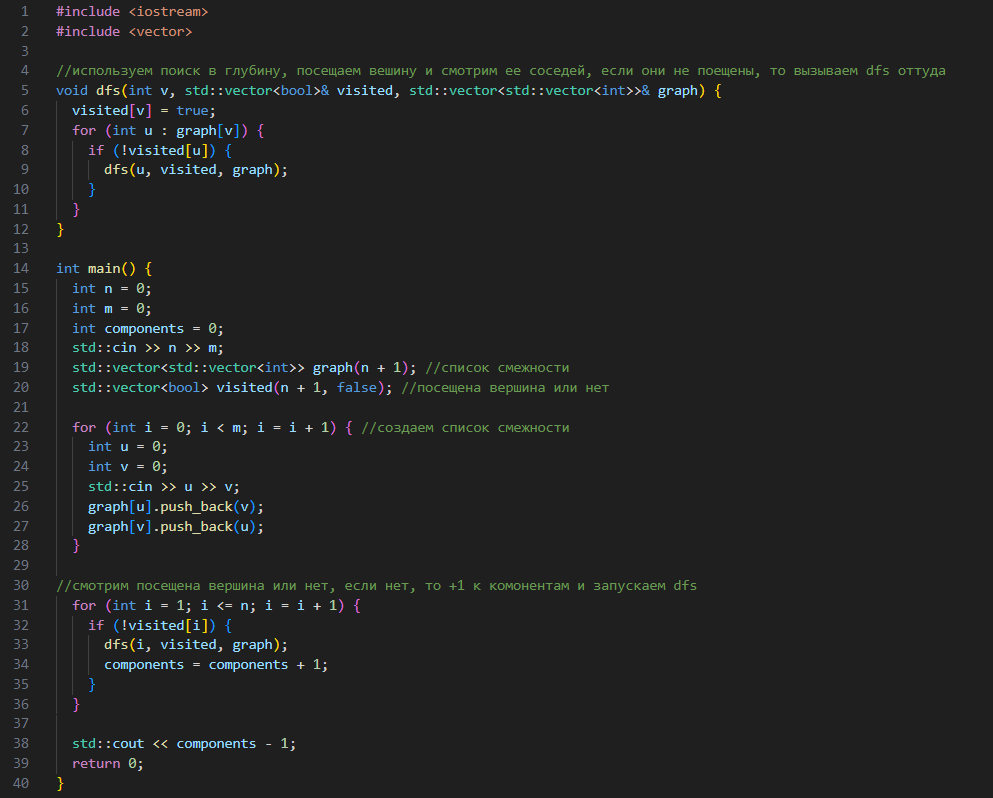
****

Рисунок 4 Код программы задачи 2

**Результаты:**

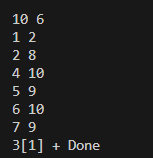


Рисунок 5 Результат программы задачи 2

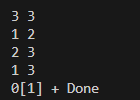


Рисунок 6 Результат программы задачи 2

**Задача №3**

В Иркутске раз в году наступает зима. Несмотря на то, что событие это довольно регулярное, оно всегда внезапно. Снег буквально заваливает все улицы, не давая проехать на чём-то меньше трактора. В этом году терпение лопнуло и специальным указом был создан кризисный центр по борьбе с сугробами. Центру были переданы спутники, лазеры, метеорологические зонды и несколько десятков лопат. Вам поручено возглавить отдел разведки снежной ситуации и быть способным чрезвычайно быстро отвечать на запросы центра. Сам город состоит из нескольких, расположенных подряд, улиц, каждая из которых абсолютна похожа на любую другую. Информация о снеге передается вам в виде тройки чисел – 1 в качестве идентификатора события, уникального индекса улицы и количество миллиметров выпавшего снега. Запросы в свою очередь так же имеют вид тройки чисел – 2 в качестве идентификатора события, индекс улицы с которой нужно суммировать количество выпавшего снега и индекс улицы, по которую нужно суммировать, крайние улицы должны быть включены.

**Алгоритм программы:**

**1. Инициализация:** читаем количество улиц n и запросов k. Создаём массив amount\_precipitation для хранения снега на каждой улице (индексация с 0). Создаём массив sum\_arr для хранения результатов запросов.

**2. Обработка запросов:** для каждого запроса: если тип 1 (снегопад) - увеличиваем количество снега на указанной улице (amount\_precipitation[i] += x. Если тип 2 (запрос суммы) - суммируем снег в диапазоне улиц [u, r] и сохраняем результат в sum\_arr.

**3. Вывод результатов:** выводим все сохранённые результаты запросов типа 2.

**4. Освобождение памяти:** удаляем динамические массивы.

**Код программы:**

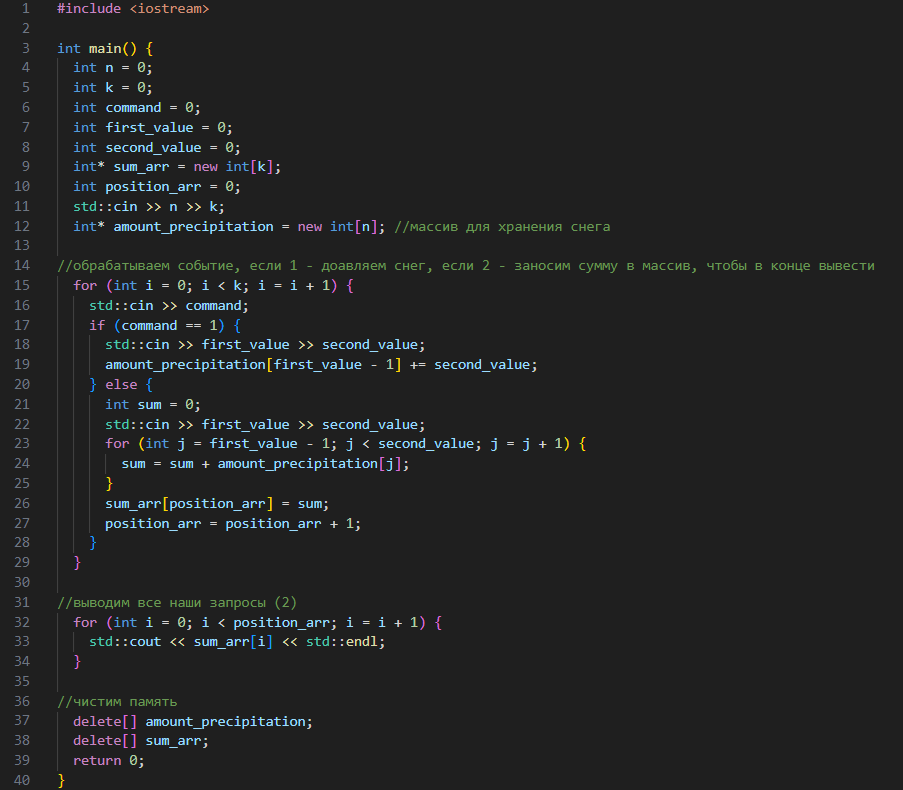
****

Рисунок 7 Код программы задачи 3

**Результаты:**

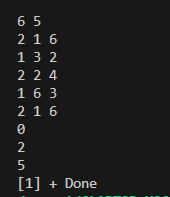
****

Рисунок 8 Результат программы задачи 3

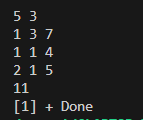
****

Рисунок 9 Результат программы задачи 3

**Задача №4**

Перестановка P длины n — это упорядоченный набор, содержащий числа от 1 до n, каждое из которых входит в него ровно один раз. Например, перестановкой длины 13 является набор (5 11 13 12 6 1 8 4 10 9 7 2 3). Само название говорит о том, для чего предназначен этот объект. Например, можно при помощи перестановки букв зашифровать слово. Для примера возьмем приведенную выше перестановку и слово transposition, которое состоит тоже из 13 букв. Далее, следуя перестановке, на первую позицию поставим пятую букву слова, на вторую − одиннадцатую букву и так далее. В итоге получим sinoptsntiora. К этому слову снова применим эту же перестановку и получим poartsnoitsin. Повторив эти стадии шифрования k раз, получим зашифрованное сообщение. Вам дано зашифрованное таким образом слово, шифрующая перестановка P и число k. Необходимо восстановить слово.

**Алгоритм программы:**

1.**Инициализация:** чтение входных данных: длины перестановки n, количества применений k, самой перестановки permutation и зашифрованного слова encrypted\_word. Создание копии зашифрованного слова original\_word для восстановления исходного сообщения.

2.**Обратное преобразование:** цикл по количеству применений k: для каждой позиции j в перестановке: переносим букву из encrypted\_word[j] в позицию permutation[j] - 1 в original\_word (обратное действие шифрованию). Обновляем encrypted\_word значением original\_word для следующей итерации.

3.**Вывод результата:** после k итераций original\_word содержит исходное слово, которое выводится на экран.

**Код программы:**

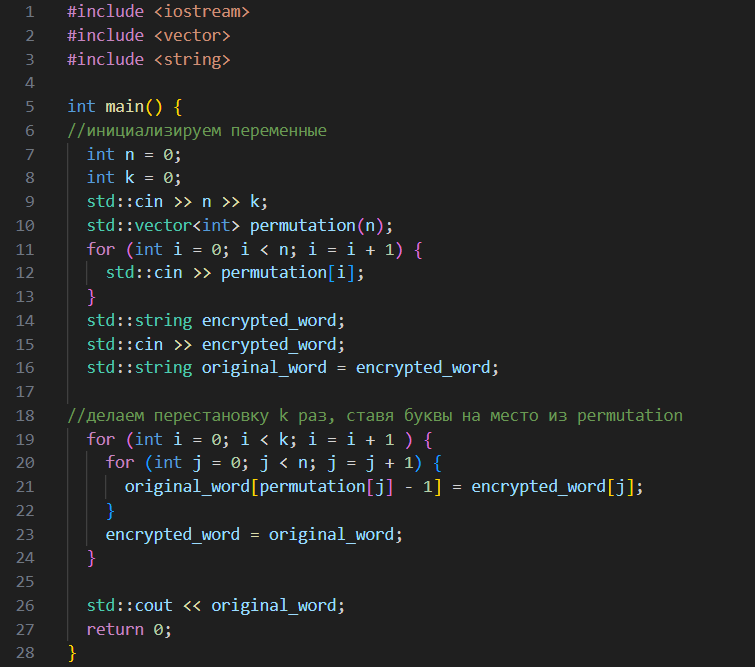


Рисунок 10 Код программы задачи 4

**Результаты:**

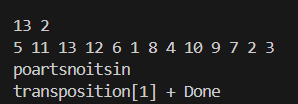


Рисунок 11 Результат программы задачи 4

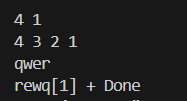


Рисунок 12 Результат программы задачи 4

**Задача №5**

Дана матрица, состоящая из 1 и 0. Значениями 1 в матрице нарисована некоторая фигура. Необходимо определить координаты верхнего левого и нижнего правого углов параллельного осям ограничивающего прямоугольника, т.е. такого прямоугольника, минимального размера, в который фигура помещается полностью и при этом ни одна точка исходной фигуры не попадает на стороны прямоугольника.

**Алгоритм программы:**

1.**Инициализация:** чтение размеров матрицы h и w. Заполнение матрицы значениями 0 или 1.

2.**Поиск границ фигуры:** проход по всем элементам матрицы. Для каждого элемента со значением 1 обновляются границы: top — минимальная строка с 1. left — минимальный столбец с 1. bottom — максимальная строка с 1. right — максимальный столбец с 1.

3.**Вывод результата:** координаты ограничивающего прямоугольника вычисляются как: верхний левый угол: (top - 1, left - 1), нижний правый угол: (bottom + 1, right + 1).

**Код программы:**

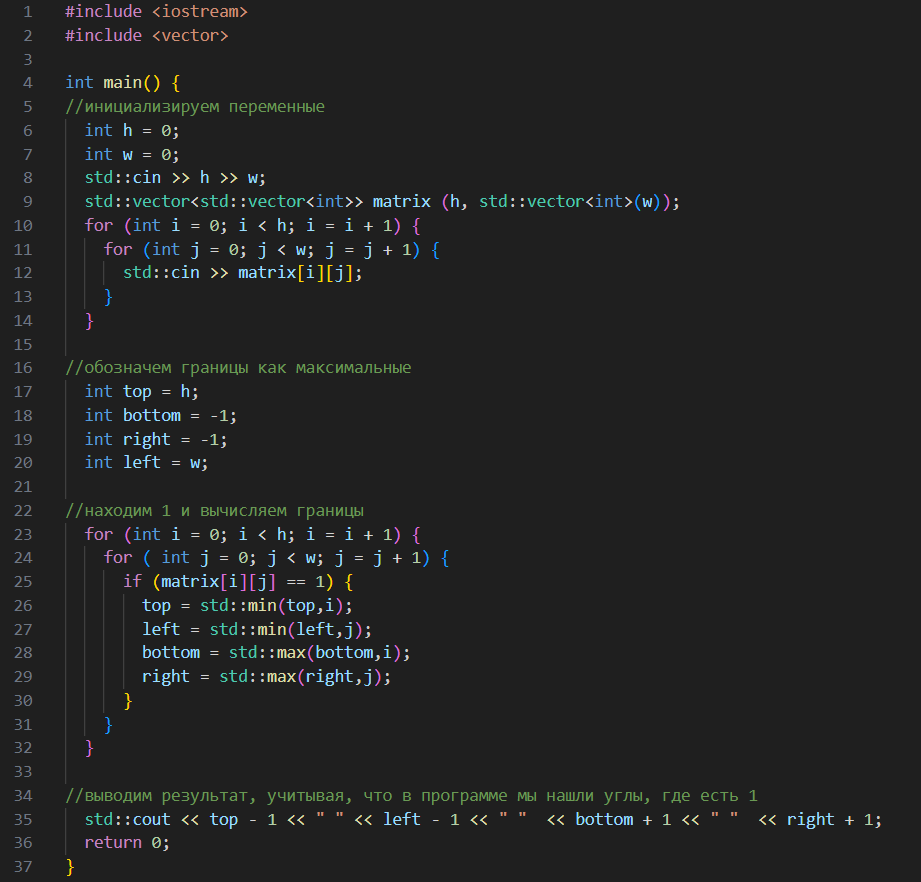
****

Рисунок 13 Код программы задачи 5

**Результаты:**

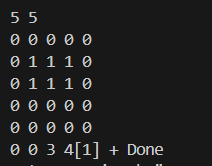
****

Рисунок 14 Результат программы задачи 5

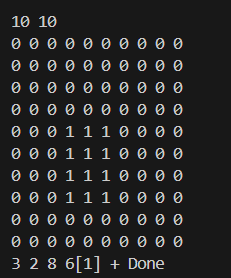
****

Рисунок 15 Результат программы задачи 5

**Задача №6**

В школьном кружке робототехники есть два вида микроконтроллеров (условно тип A и тип B) и два вида модулей управления мотором (условно тип 1 и тип 2). Выяснилось, что контроллер типа B и модуль управления типа 2 несовместимы. Использование микроконтроллеров и модулей управления в других комбинациях возможно. Имеется a микроконтроллеров типа A, b микроконтроллеров типа B, x модулей управления типа 1 и y модулей типа 2. Определите, какое максимальное число работающих пар из микроконтроллера и модуля управления мотором можно составить. Ваша программа должна ответить на n запросов.

**Алгоритм программы:**

1.**Инициализация:** чтение количества запросов n. Создание массива result для хранения результатов каждого запроса.

2.**Обработка запросов:** для каждого запроса считываются значения a, b, x, y: b\_x — количество пар (B, 1), ограничено минимумом из b и x. a\_x — количество пар (A, 1), оставшихся после использования b\_x, ограничено минимумом из a и оставшихся x - b\_x. a\_y — количество пар (A, 2), ограничено минимумом из оставшихся a - a\_x и y. Результат для запроса: сумма b\_x + a\_x + a\_y.

3.**Вывод результатов:** вывод всех результатов из массива result через пробел.

**Код программы:**

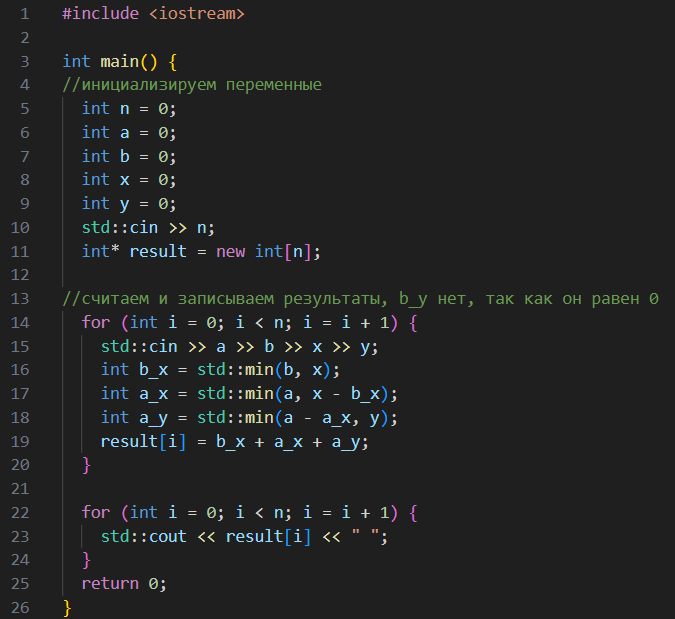
****

Рисунок 16 Код программы задачи 6

**Результаты:**

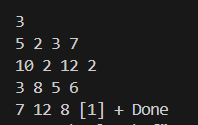
****

Рисунок 17 Результат программы задачи 6

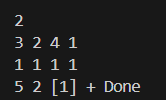


Рисунок 18 Результат программы задачи 6

**Задача №7**

На компьютере работника автосервиса нашли файл с последовательностью автомобильных номеров, обслуживавшихся в этом автосервисе. Так как файл был поврежден, некоторые данные отображаются неверно. Нужно определить, какие из них остались невредимыми. Автомобильным номером является строка из шести символов. Первый символ – заглавная латинская буква, далее следует 3 цифры, и после – две заглавные латинские буквы. Например, строка "P142EQ" является номером. Вам будет дана строка, состоящая из шести символов, необходимо ответить, является ли строка автомобильным номером.

**Алгоритм программы:**

1.**Инициализация:** создаются строки letters (все заглавные латинские буквы) и numbers (все цифры). Считывается строка car\_number для проверки.

2.**Проверка формата номера:** символы 1, 5-6**:** должны быть буквами (проверяется наличие в letters). Символы 2-4**:** должны быть цифрами (проверяются через numbers).

3.**Вывод результата:** если все условия выполнены - выводится "YES", если хотя бы одно условие не выполняется, то выводится "NO".

**Код программы:**

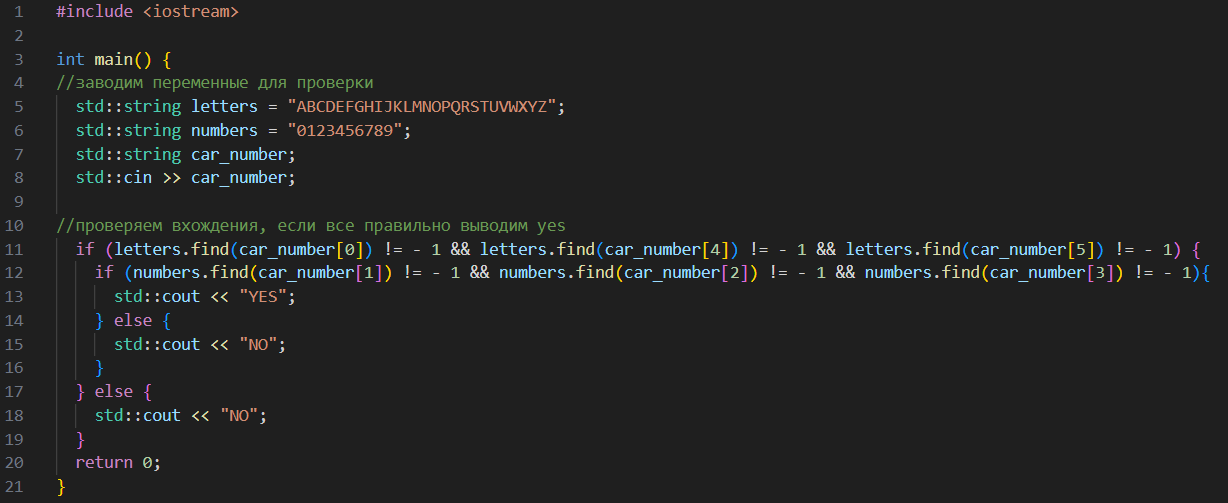
****

Рисунок 19 Код программы задачи 7

**Результаты:**



Рисунок 20 Результат программы задачи 7



Рисунок 21 Результат программы задачи 7

**Задача №8**

Составить светодиодную матрицу размером не менее 8 на 8 светодиодов (пример на рисунке ниже размером 4 на 4). На матрицу вывести инфографику с различными динамично меняющимися изображениями.

**Алгоритм программы:**

1.**Инициализация:** подключается библиотека LedControl для управления матрицей. Указываются пины Arduino для подключения (DIN, CLK, CS). Создается объект lc для управления матрицей.

2.**Хранение изображений:** в массиве heartPatterns хранятся 3 варианта изображения сердца.

3.**Настройка матрицы setup():** включается матрица shutdown(0, false). Устанавливается яркость setIntensity(0, 10). Очищается дисплей clearDisplay(0).

4.**Бесконечный цикл анимации loop():** поочередно отображаются все 3 варианта сердца с задержкой 400 мс между сменами.

5.**Функция showHeart():** принимает индекс изображения patternIndex. Загружает построчно данные из heartPatterns в матрицу с помощью lc.setRow().

**Код программы:**

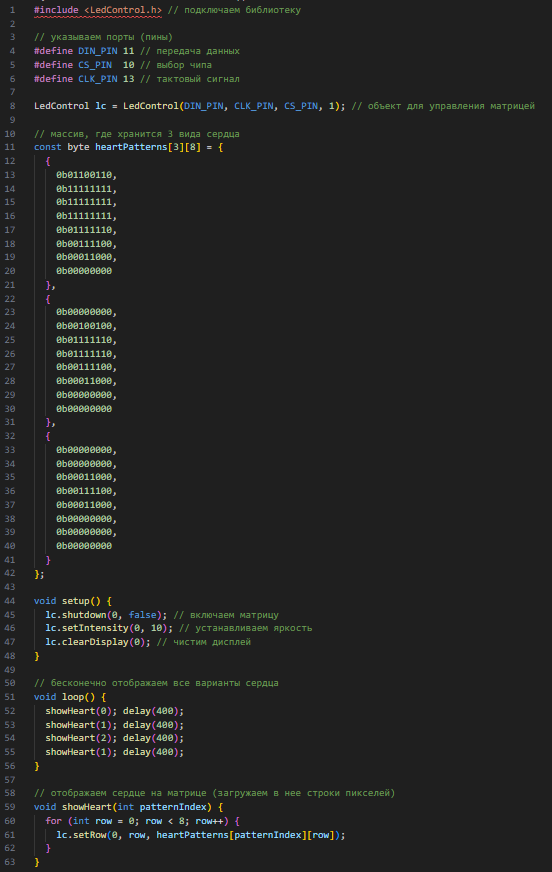


Рисунок 22 Код программы задачи 8

**Результаты:**

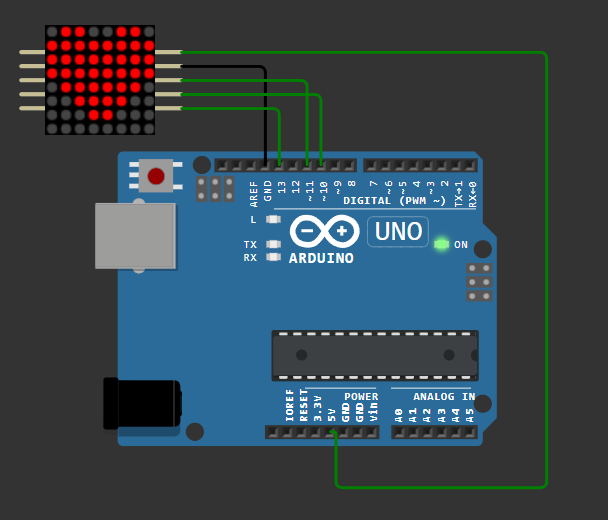
****

Рисунок 23 Результат программы задачи 8

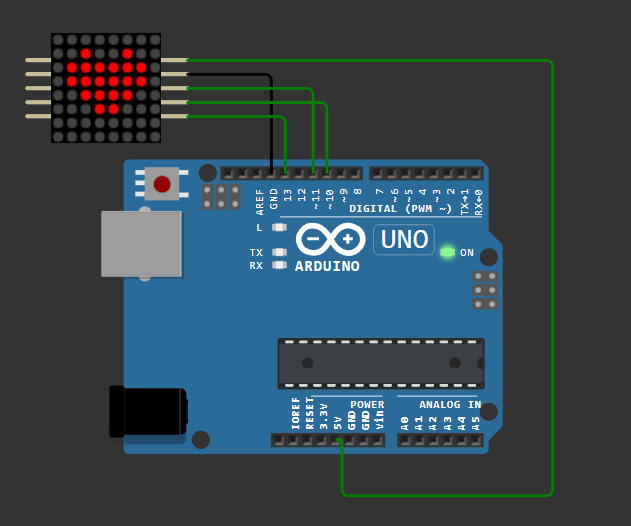
****

Рисунок 24 Результат программы задачи 8

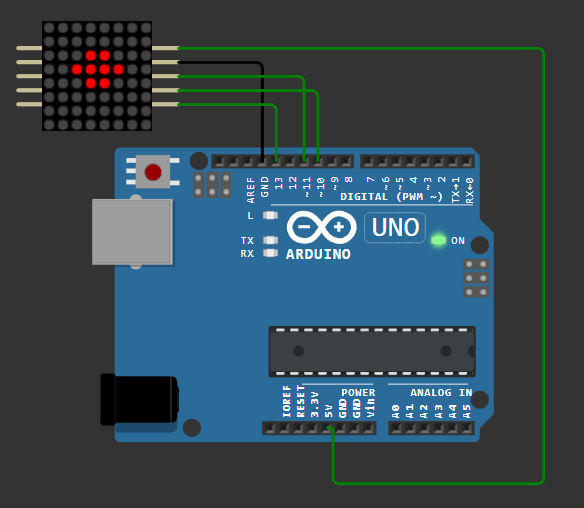


Рисунок 25 Результат программы задачи 8

**Задача №9**

Собрать схему имитирующую работу автоматических дверей. По умолчанию горит светодиод, имитирующий закрытую дверь. Микроконтроллер считывает значение фоторезистора с аналогово пина. Если значение на пине превышает 512, на определённое время загорается светодиод, имитирующий открытую дверь, в последовательный порт выводится сообщение о событии. После истечения заданного временного промежутка проверяется значение фоторезистора, если оно всё ещё превышает 512, дверь должна остаться открытой, в противном случае нужно включить индикацию закрытой двери, в последовательный порт выводится сообщение о событии.

**Алгоритм программы:**

1.**Инициализация:** назначаем пины для фоторезистора, для зелёного и красного светодиодов. Устанавливаем пороговое значение THRESHOLD и время открытия DELAY\_TIME мс.

2.**Основной цикл:** читаем значение с фоторезистора analogRead(PHOTO\_PIN). Если значение > THRESHOLD: выключаем красный, включаем зелёный светодиод. Ждём DELAY\_TIME, затем снова проверяем фоторезистор. Если значение ≤ THRESHOLD, выключаем зелёный, включаем красный. Сообщения о состоянии дверей выводятся.

**Код программы:**

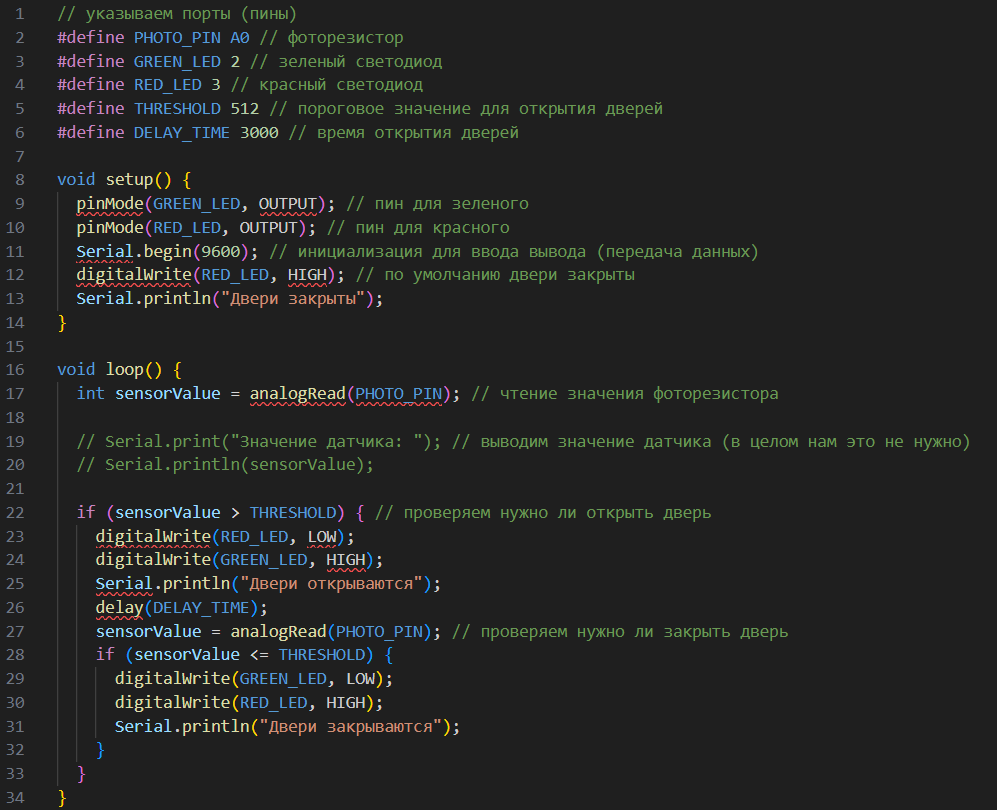
****

Рисунок 26 Код программы задачи 9

**Результаты:**

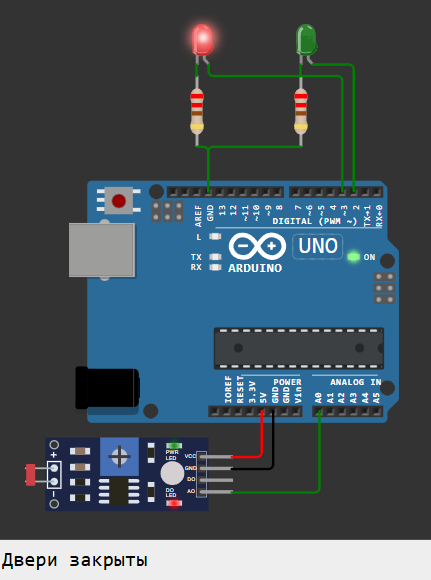
****

Рисунок 27 Результат программы задачи 9

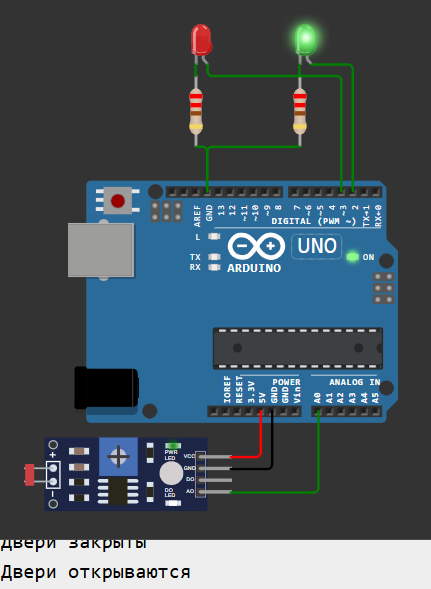
****

Рисунок 28 Результат программы задачи 9

**Задача №10**

Собрать схему подключения сервопривода. Написать программу для управления сервоприводом через последовательный порт. Программа находится в ожидании ввода данных в последовательный порт. Когда в последовательный порт вводятся данные, программа проверяет их на корректность (Должны приниматься только числовые значения в диапазоне от 0 до 180). Если введён угол поворота, отличный от текущего угла, сервопривод плавно поворачивается в заданное положение.

**Алгоритм программы:**

1.**Инициализация:** подключаем библеотеку, пины, создаем объект myServo для управления.

2.**Основной цикл:** проверяем наличие данных в Serial-порту. Если данные есть: читаем строку до символа \n и удаляем пробелы trim(). Проверяем, является ли ввод числом isValidNumber(). Если число корректное: плавно поворачиваем сервопривод от currentAngle к targetAngle шаг 1°. Обновляем currentAngle после завершения поворота. Если число некорректное — выводим сообщение об ошибке.

3.**Функция isValidNumber:** проверяет, что каждый символ строки — цифра.

**Код программы:**

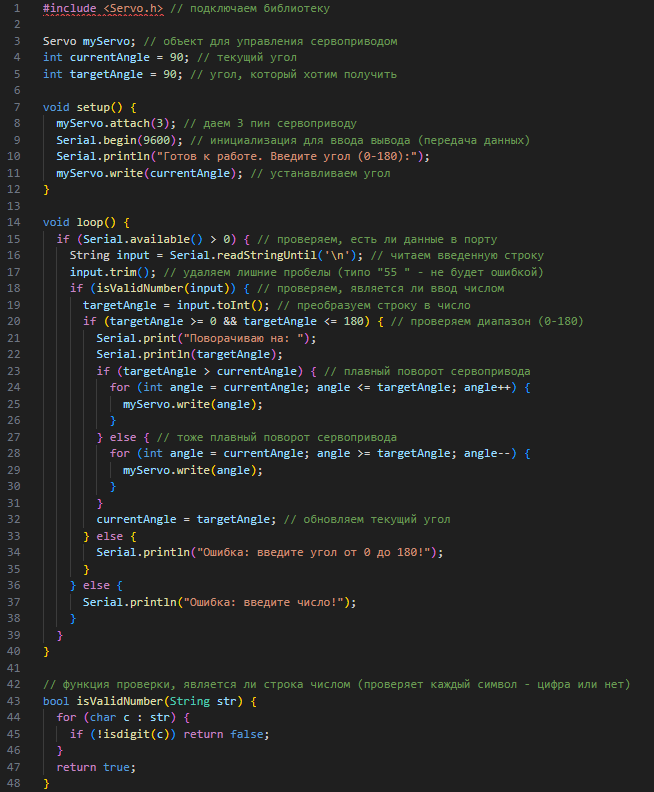
****

Рисунок 29 Код программы задачи 10

**Результаты:**

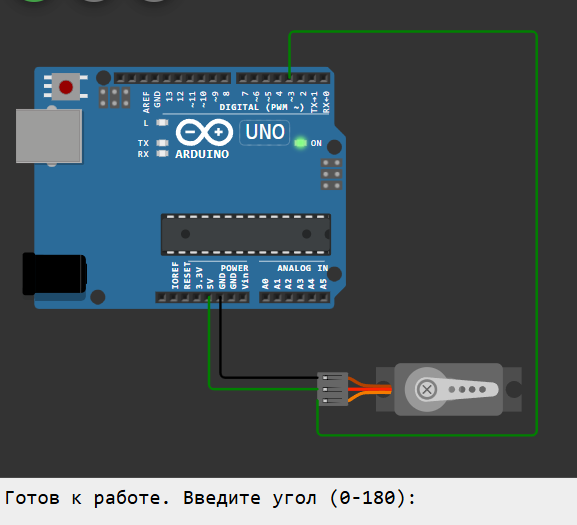
****

Рисунок 30 Результат программы задачи 10

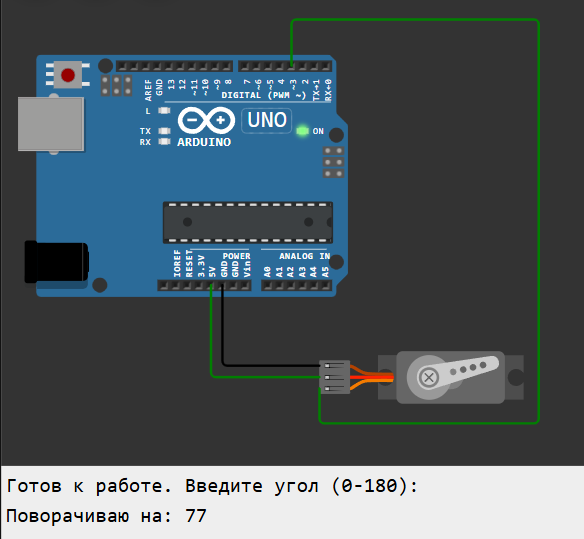
****

Рисунок 31 Результат программы задачи 10

**Задача №11**

Find all green objects in the image. Find the centers of green objects. Mark the center with a red dot.

Найдите все зеленые объекты на изображении. Найдите центры зеленых объектов. Отметьте центр красной точкой.

**Алгоритм программы:**

1.**Загрузка изображения:** чтение файла test2.jpg и отображение исходного изображения.

2.**Предварительная обработка:** применение размытия Гаусса для сглаживания шумов. Преобразование цветового пространства из BGR в HSV.

3.**Создание маски:** определение диапазона зеленого цвета в HSV hsv\_min и hsv\_max. Создание бинарной маски, где белым выделены зеленые объекты.

4.**Поиск контуров:** обнаружение контуров зеленых объектов на маске. Для каждого контура: вычисление координат ограничивающего прямоугольника. Определение центра объекта center\_x и center\_y. Рисование красной точки в центре каждого объекта.

5.**Отображение результатов:** показ изображения с отмеченными центрами зеленых объектов. Закрытие всех окон по нажатию клавиши.

**Код программы:**

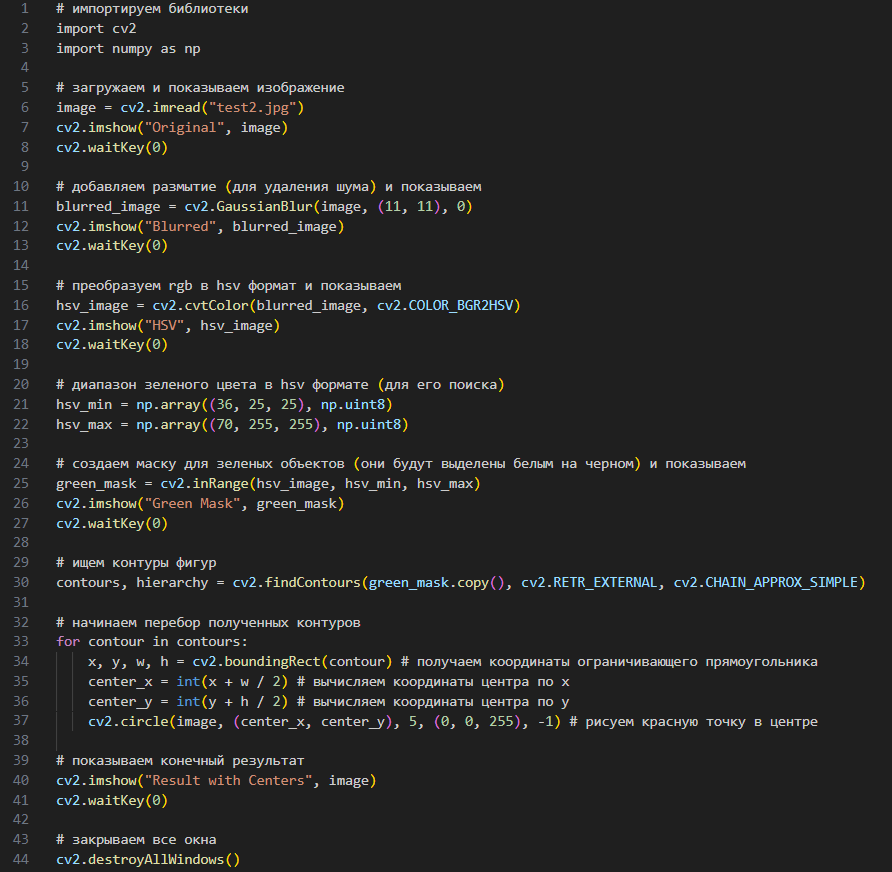
****

Рисунок 32 Код программы задачи 11

**Результаты:**

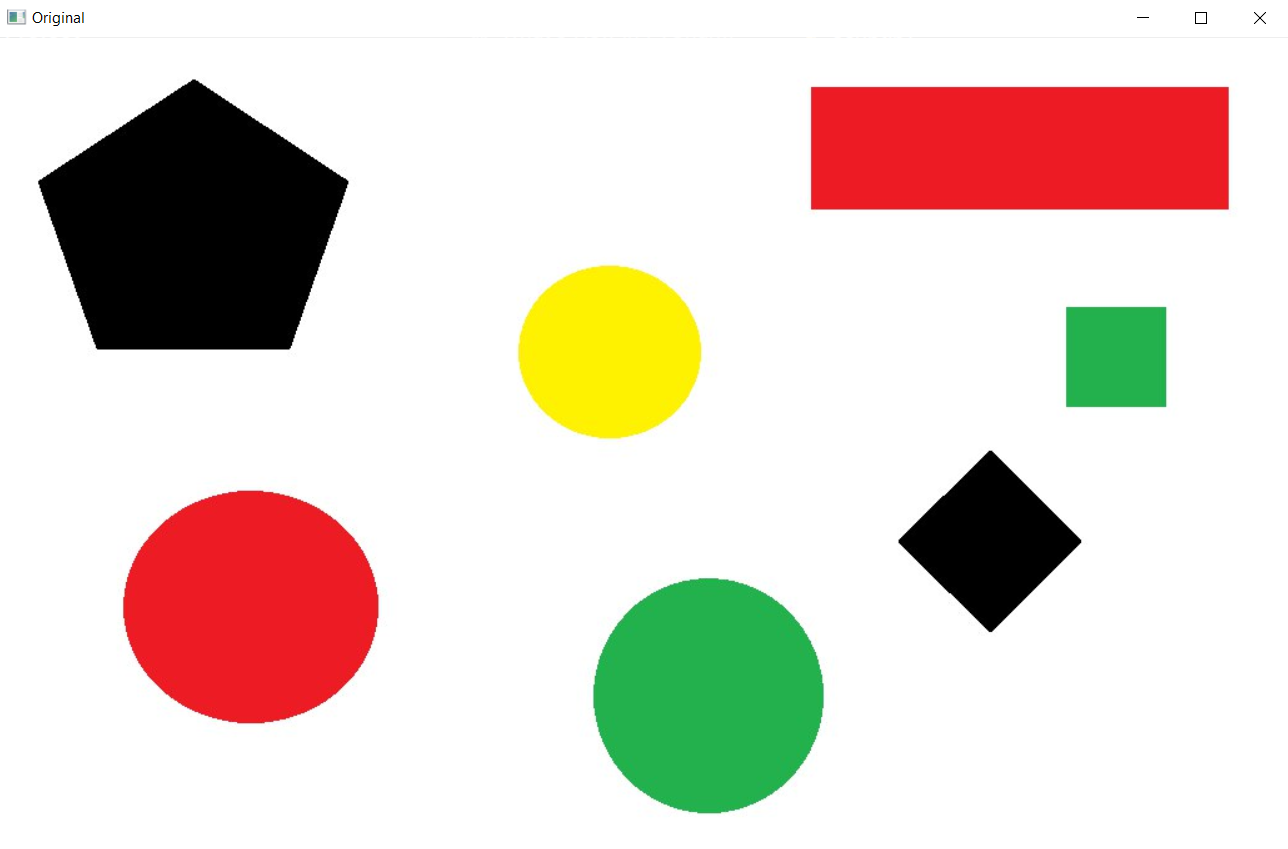


Рисунок 33 Результат программы задачи 11



Рисунок 34 Результат программы задачи 11

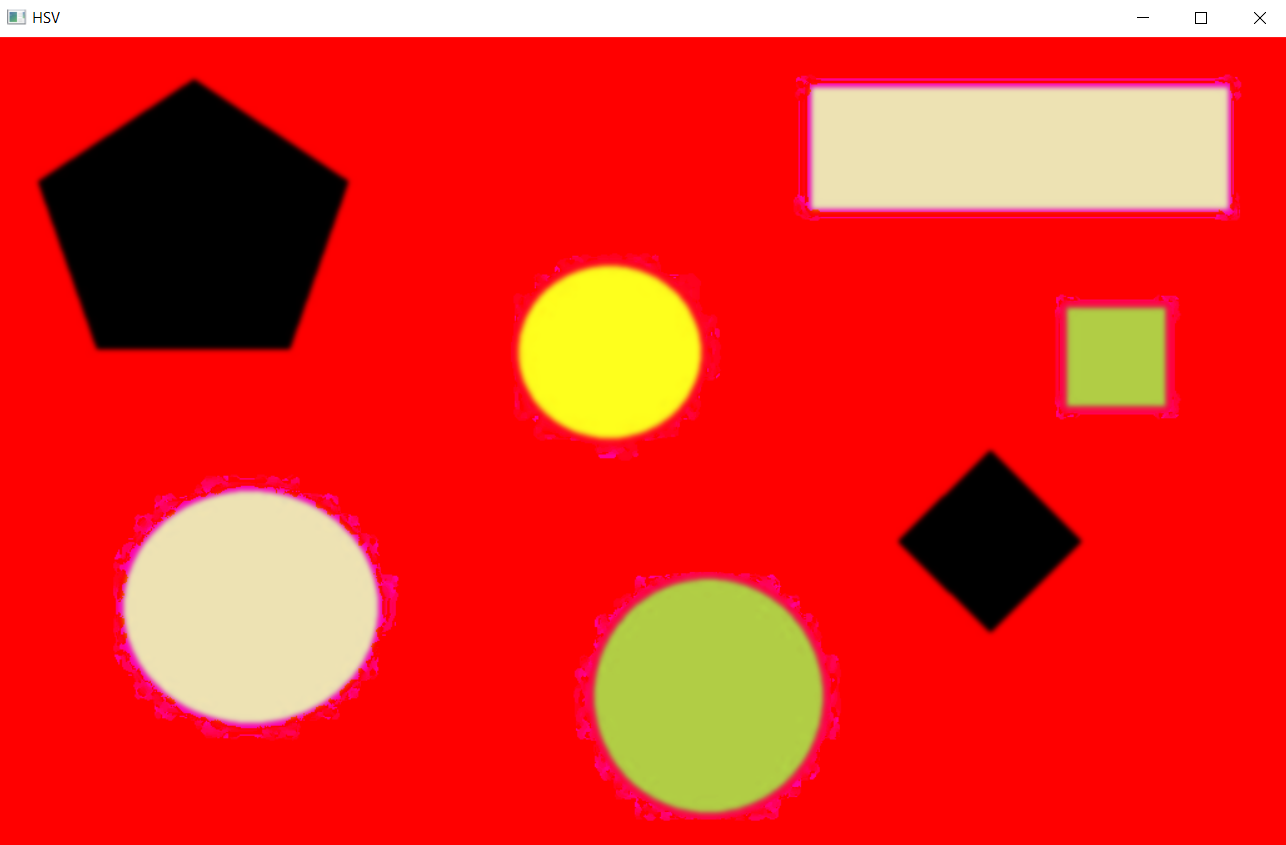


Рисунок 35 Результат программы задачи 11

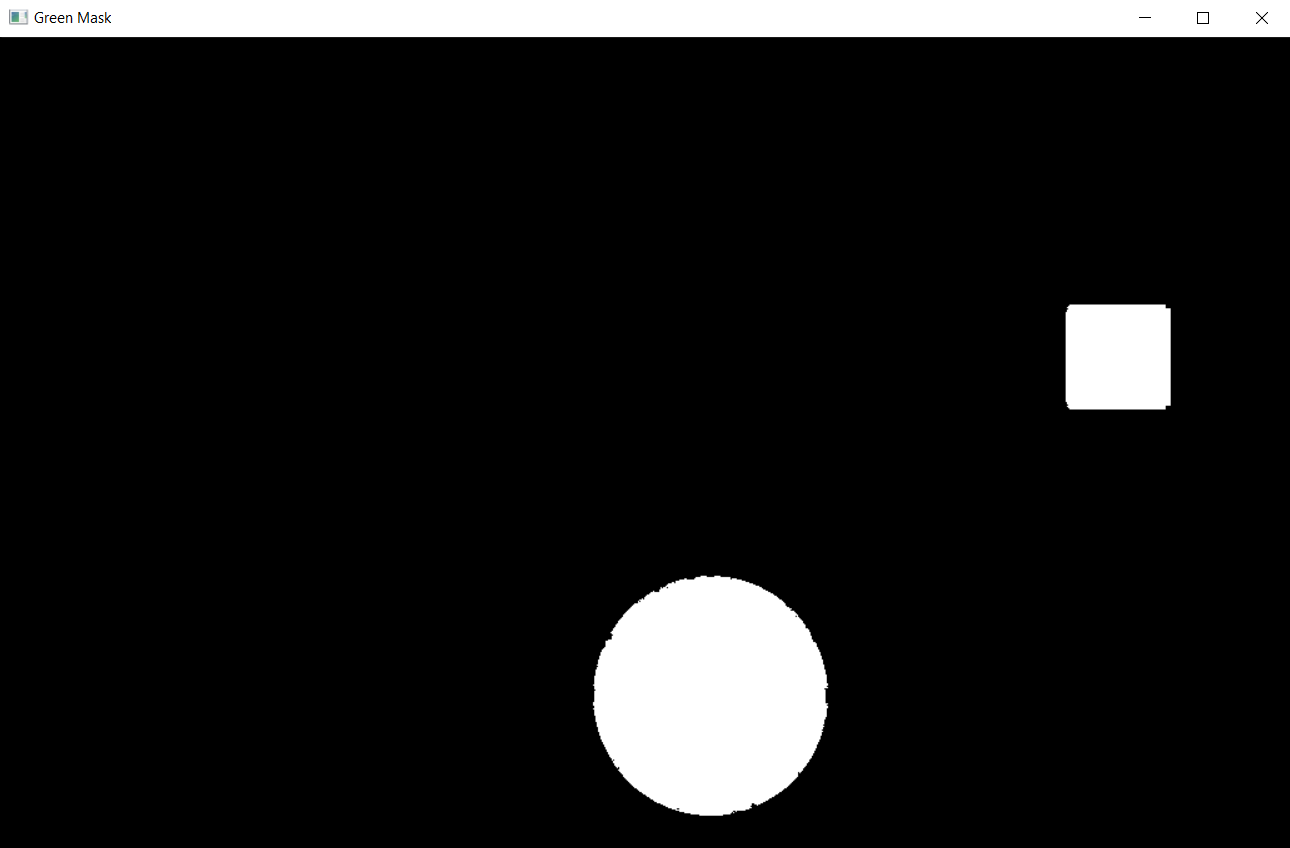


Рисунок 36 Результат программы задачи 11

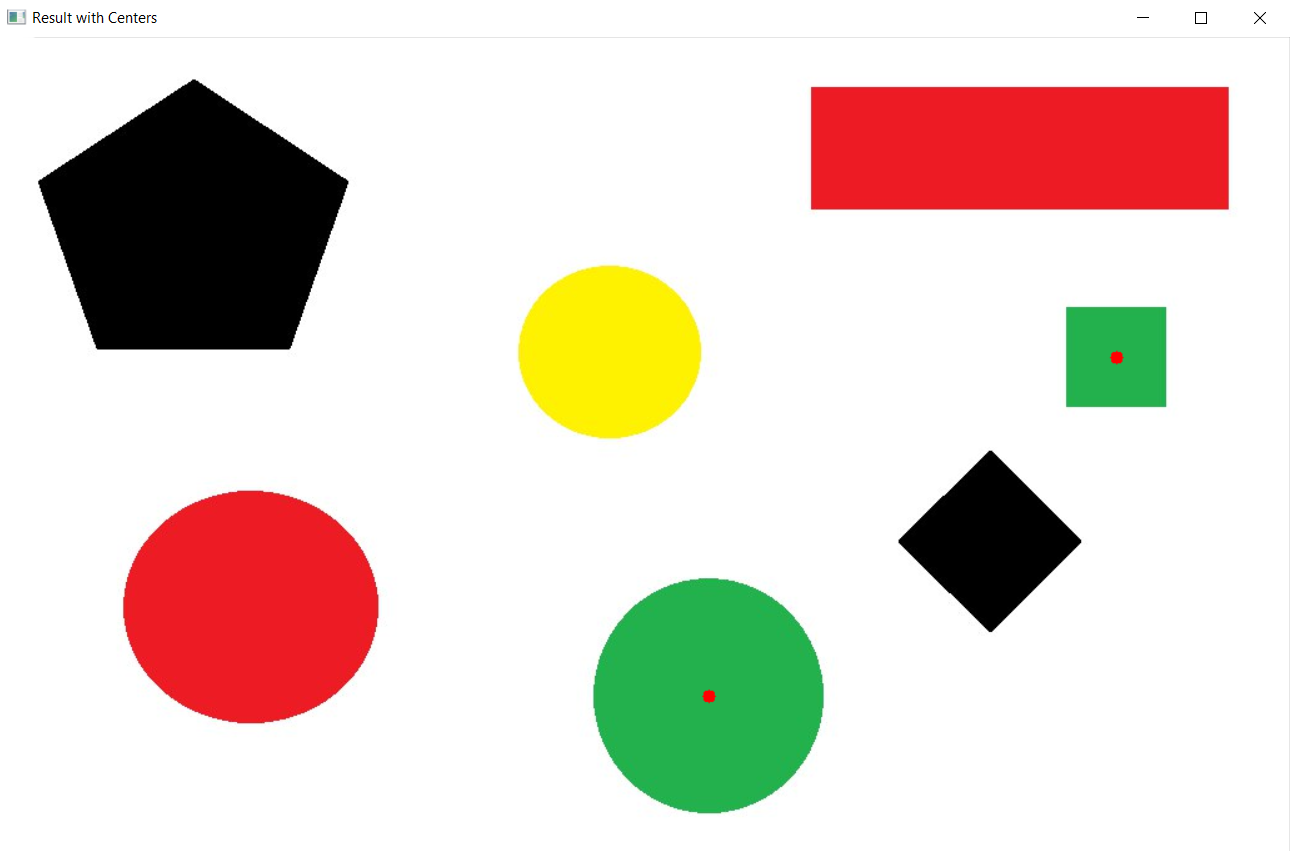


Рисунок 37 Результат программы задачи 11