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 Василец В. Р., ИСИб-24-1

(ФИО, группа, подпись)

Руководитель практики от института ИТиАД

Кононенко Роман Владимирович, доцент

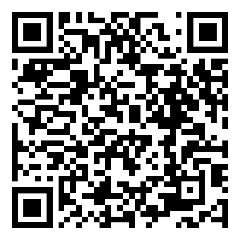
Ссылка на резюме https://www.superjob.ru/resume/programmist-s-55733621.html

(ФИО, должность, подпись)

Руководитель образовательной программы

Кононенко Р.В., доцент института ИТиАД

(ФИО, должность, подпись)

Оценка по практике

(ФИО, подпись, дата)

Содержание отчета на \_\_\_ стр. Приложение к отчету на \_\_\_ стр

Ссылка на резюме https://irkutsk.hh.ru/resume/b26a6c3eff0efde0e50039ed1f61686c6b4d49

Иркутск 2025

**Индивидуальное задание на прохождение**

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| для | Васильца Василия Романовича | | | | |
|  | (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: улучшить навыки программирования, ознакомится с деятельностью IT-компаний

Содержание практики, вопросы, подлежащие изучению: решение задач на логику, написание программ для электронных устройств, разработка ПО, посещение IT-организаций, которые располагаются в городе Иркутск

Планируемые результаты практики: отчет по учебной практике

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

\_\_\_\_\_\_\_\_\_\_«16» июня 2025 г.

(подпись)

**ДНЕВНИК**

прохождения практики

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| обучающегося | | | | | Васильца Василия Романовича , 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

(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 | Выполнены задачи №1, №2, №3 |  |
| 2 | 18.06.2025 | Создал резюме на сайтах hh.ru и superjob |  |
| 3 | 20.06.2025 | Выполнены задачи №4, №5, №6 |  |
| 4 | 24.06.2025 | Посетил АО «СО ЕЭС» Иркутское РДУ |  |
| 6 | 25.06.2025 | Выполнены задачи №7, №8, №9 |  |
| 7 | 26.06.2025 | Посетил компанию «ISPsystem» |  |
| 8 | 27.06.2025 | Выполнены задачи №10, №11 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |
| --- | --- |
| Дата фактического прибытия |  |
| обучающегося в структурное подразделение | 16.06.2026 |
| 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) |

**Оглавление**

[**Задание №1** 6](#_Toc209699916)

[**Задание №2** 8](#_Toc209699917)

[**Задание №3** 9](#_Toc209699918)

[**Задание №4** 11](#_Toc209699919)

[**Задание №5** 12](#_Toc209699920)

[**Задание №6** 14](#_Toc209699921)

[**Задание №7** 15](#_Toc209699922)

[**Задание №8** 16](#_Toc209699923)

[**Задание №9** 18](#_Toc209699924)

[**Задание №10** 21](#_Toc209699925)

[**Задание №11** 25](#_Toc209699926)

[**Вывод** 27](#_Toc209699927)

# **Задание №1**

Постановка задачи:

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. Напишите программу, рассчитывающую минимальное время спуска (сумму чисел в пути с вершины до основания).

Ход решения:

**import** **random**

**def** **time**(n, pyramid):

dp = [[**0**] \* (i + **1**) **for** i **in** range(n)]

dp[**0**][**0**] = pyramid[**0**][**0**]

**for** i **in** range(**1**, n):

**for** j **in** range(i + **1**):

**if** j == **0**:

dp[i][j] = dp[i - **1**][j] + pyramid[i][j]

**elif** j == i:

dp[i][j] = dp[i - **1**][j - **1**] + pyramid[i][j]

**else**:

dp[i][j] = min(dp[i - **1**][j - **1**], dp[i - **1**][j]) + pyramid[i][j]

min\_time = min(dp[n - **1**])

**return** min\_time, dp

**def** **path**(n, pyramid, dp):

min\_index = dp[n - **1**].index(min(dp[n - **1**]))

path = []

current\_index = min\_index

**for** i **in** range(n - **1**, -**1**, -**1**):

path.append(pyramid[i][current\_index])

**if** i > **0**:

**if** current\_index > **0** **and** dp[i - **1**][current\_index - **1**] < dp[i - **1**][current\_index]:

current\_index -= **1**

path.reverse()

**return** path

n = int(input())

pyramid = []

**for** i **in** range(n):

level = [random.randint(**1**, **100**) **for** \_ **in** range(i + **1**)]

pyramid.append(level)

**for** level **in** pyramid:

**print**(" ".join(map(str, level)))

time, dp = time(n, pyramid)

path = path(n, pyramid, dp)

**print**("Минимальное время пути:", time)

**print**("Сам путь", " ".join(map(str, path)))

Результат:

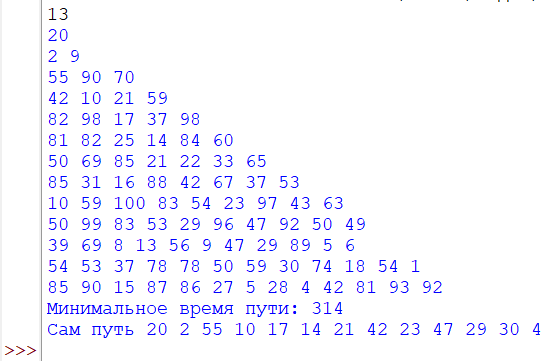


Рисунок 1 – пример выполнения программы 1

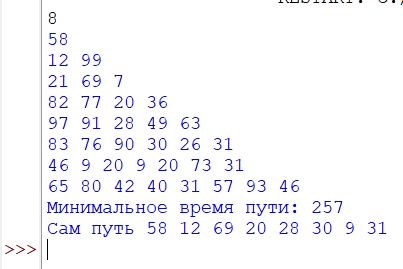


Рисунок 2 – пример выполнения программы 2

# **Задание №2**

Постановка задачи:

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.

Ход решения:

**def** **components**(n, e):

graph = {i: [] **for** i **in** range(**1**, n + **1**)}

**for** u, v **in** e:

graph[u].append(v)

graph[v].append(u)

visited = set()

components = **0**

**def** **dfs**(node):

stack = [node]

**while** stack:

curr = stack.pop()

**for** neighbor **in** graph[curr]:

**if** neighbor **not** **in** visited:

visited.add(neighbor)

stack.append(neighbor)

**for** node **in** range(**1**, n + **1**):

**if** node **not** **in** visited:

components += **1**

visited.add(node)

dfs(node)

**return** components - **1**

n, m = map(int, input().split())

e = [tuple(map(int, input().split())) **for** \_ **in** range(m)]

**print**(components(n, e))

Результат:

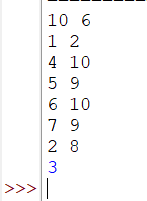


Рисунок 3 – пример выполнения программы 1

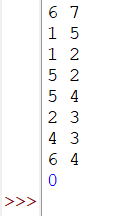


Рисунок 4 – пример выполнения программы 2

# **Задание №3**

Постановка задачи:

В Иркутске раз в году наступает зима. Не смотря 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.

Ход решения:

**def** **snow**(n, k, events):

snow = [**0**] \* (n + **1**)

results = []

**for** event **in** events:

event\_type = event[**0**]

**if** event\_type == **1**:

street\_index = event[**1**]

snow\_amount = event[**2**]

snow[street\_index] += snow\_amount

**elif** event\_type == **2**:

start\_index = event[**1**]

end\_index = event[**2**]

total\_snow = sum(snow[start\_index:end\_index + **1**])

results.append(total\_snow)

**return** results

**if** \_\_name\_\_ == "\_\_main\_\_":

n, k = map(int, input().split())

events = [list(map(int, input().split())) **for** \_ **in** range(k)]

results = snow(n, k, events)

**for** result **in** results:

**print**(result)

Результат:

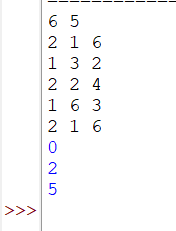


Рисунок 5 – пример выполнения программы 1

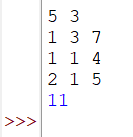


Рисунок 6 – пример выполнения программы 2

# **Задание №4**

Постановка задачи:

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.

Ход решения:

**def** **inverse**(permutation):

n = len(permutation)

inv = [**0**] \* n

**for** i **in** range(n):

inv[permutation[i] - **1**] = i + **1**

**return** inv

**def** **decrypt**(encrypted\_word, permutation, k):

n = len(encrypted\_word)

inv = inverse(permutation)

current\_word = encrypted\_word

**for** \_ **in** range(k):

current\_word = ''.join(current\_word[inv[i] - **1**] **for** i **in** range(n))

**return** current\_word

n, k = map(int, input().split())

permutation = list(map(int, input().split()))

encrypted\_word = input().strip()

original\_word = decrypt(encrypted\_word, permutation, k)

print(original\_word)

Результат:

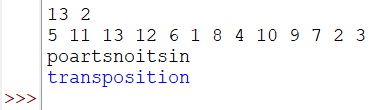


Рисунок 7 – пример выполнения программы 1

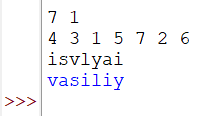


Рисунок 8 – пример выполнения программы 2

# **Задание №5**

Постановка задачи:

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.

Ход решения:

**def** **rectangle**(matrix, h, w):

min\_row, min\_col = h, w

max\_row, max\_col = -**1**, -**1**

**for** i **in** range(h):

**for** j **in** range(w):

**if** matrix[i][j] == **1**:

**if** i < min\_row:

min\_row = i

**if** j < min\_col:

min\_col = j

**if** i > max\_row:

max\_row = i

**if** j > max\_col:

max\_col = j

**return** min\_row, min\_col, max\_row, max\_col

h, w = map(int, input().split())

matrix = [list(map(int, input().split())) **for** \_ **in** range(h)]

top\_left\_row, top\_left\_col, bottom\_right\_row, bottom\_right\_col = rectangle(matrix, h, w)

print(top\_left\_row - **1**, top\_left\_col - **1**, bottom\_right\_row + **1**, bottom\_right\_col + **1**)

Результат:

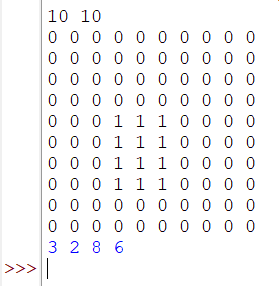


Рисунок 9 – пример выполнения программы 1

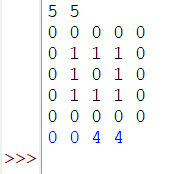


Рисунок 10 – пример выполнения программы 2

# **Задание №6**

Постановка задачи:

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.

Ход решения:

**def** **pairs**(n, test\_cases):

results = []

**for** a, b, x, y **in** test\_cases:

pairs = min(a + b, x + y, min(a, x + y) + min(b, x))

results.append(pairs)

**return** results

n = int(input())

test\_cases = [tuple(map(int, input().split())) **for** \_ **in** range(n)]

results = pairs(n, test\_cases)

print(' '.join(map(str, results)))

Результат:

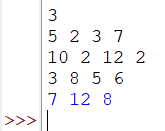


Рисунок 11 – пример выполнения программы 1

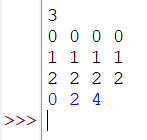


Рисунок 12 – пример выполнения программы 2

# **Задание №7**

Постановка задачи:

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.

Ход решения:

**def** **number**(plate):

**if** len(plate) != **6**:

**return** "No"

**if** **not** plate[**0**].isupper():

**return** "No"

**for** i **in** range(**1**, **4**):

**if** **not** plate[i].isdigit():

**return** "No"

**for** i **in** range(**4**, **6**):

**if** **not** plate[i].isupper():

**return** "No"

**return** "Yes"

plate = input().strip()

print(number(plate))

Результат:

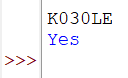


Рисунок 13 – пример выполнения программы 1

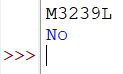


Рисунок 14 – пример выполнения программы 2

# **Задание №8**

Постановка задачи:

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 (в моем случае шашечный ряд).

Ход решения:

#include <FastLED.h>

#define PIN 3

#define WIDTH 8

#define HEIGHT 8

#define NUMPIXELS (WIDTH \* HEIGHT)

CRGB leds[NUMPIXELS];

**const** **uint8\_t** desk1[**8**][**8**] = {

{**1**,**0**,**1**,**0**,**1**,**0**,**1**,**0**},

{**1**,**0**,**1**,**0**,**1**,**0**,**1**,**0**},

{**1**,**0**,**1**,**0**,**1**,**0**,**1**,**0**},

{**1**,**0**,**1**,**0**,**1**,**0**,**1**,**0**},

{**1**,**0**,**1**,**0**,**1**,**0**,**1**,**0**},

{**1**,**0**,**1**,**0**,**1**,**0**,**1**,**0**},

{**1**,**0**,**1**,**0**,**1**,**0**,**1**,**0**},

{**1**,**0**,**1**,**0**,**1**,**0**,**1**,**0**}

};

**const** **uint8\_t** desk2[**8**][**8**] = {

{**0**,**1**,**0**,**1**,**0**,**1**,**0**,**1**},

{**0**,**1**,**0**,**1**,**0**,**1**,**0**,**1**},

{**0**,**1**,**0**,**1**,**0**,**1**,**0**,**1**},

{**0**,**1**,**0**,**1**,**0**,**1**,**0**,**1**},

{**0**,**1**,**0**,**1**,**0**,**1**,**0**,**1**},

{**0**,**1**,**0**,**1**,**0**,**1**,**0**,**1**},

{**0**,**1**,**0**,**1**,**0**,**1**,**0**,**1**},

{**0**,**1**,**0**,**1**,**0**,**1**,**0**,**1**}

};

**void** **setup**() {

FastLED.addLeds<NEOPIXEL, PIN>(leds, NUMPIXELS);

FastLED.setBrightness(**50**);

}

**void** **showchess**(**const** **uint8\_t** desk[**8**][**8**], CRGB color = CRGB::Red) {

fill\_solid(leds, NUMPIXELS, CRGB::Black);

**for** (**int** y = **0**; y < HEIGHT; y++) {

**for** (**int** x = **0**; x < WIDTH; x++) {

**if** (desk[y][x]) {

**int** idx = y \* WIDTH + x;

leds[idx] = color;

}

}

}

FastLED.show();

}

**void** **loop**() {

showchess(desk1, CRGB::Red);

delay(**1000**);

showchess(desk2, CRGB::Red);

delay(**1000**);

}

Результат:

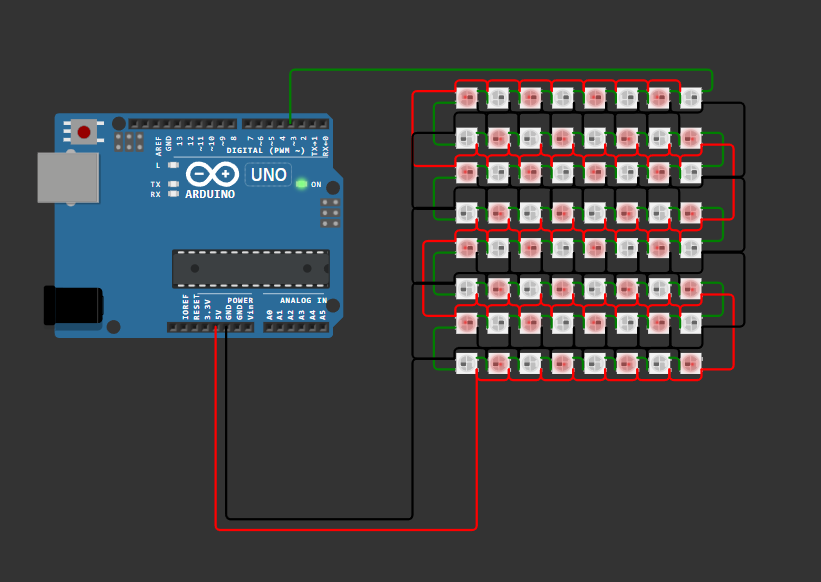
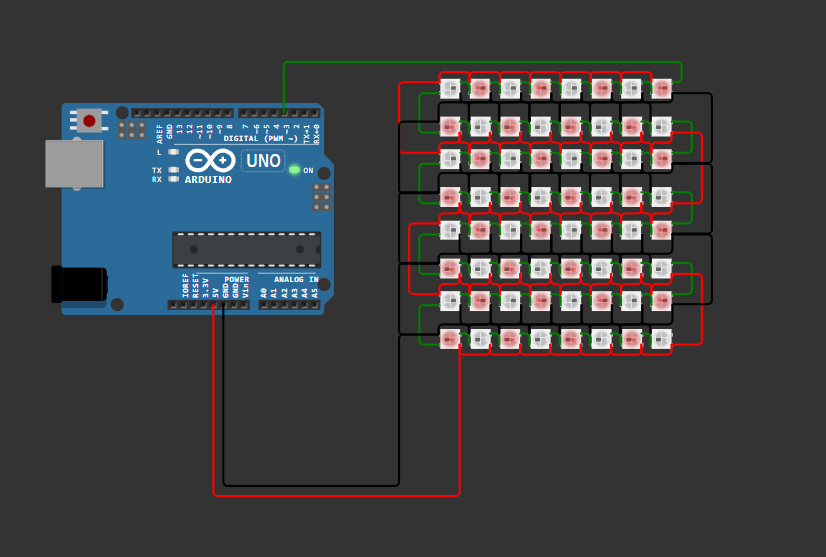


Рисунок 15 – пример выполнения программы 1

Рисунок 16 – пример выполнения программы 2

# **Задание №9**

Постановка задачи:

1. Собрать схему имитирующую работу автоматических дверей
2. Подобрать номинал резисторов для светодиодов
3. Написать программу для управления процессом работы автоматических дверей.

Логика работы программы:

1. По умолчанию горит светодиод, имитирующий закрытую дверь
2. Микроконтроллер считывает значение фоторезистора с аналогово пина
3. Если значение на пине превышает 512, на определённое время загорается светодиод, имитирующий открытую дверь, в последовательный порт выводится сообщение о событии.
4. 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.

Ход решения:

**const** **int** photoresistorPin = A0;

**const** **int** redLedPin = **9**;

**const** **int** greenLedPin = **5**;

**const** **int** threshold = **512**;

**const** **unsigned** **long** openDuration = **3000**;

**void** **setup**() {

pinMode(redLedPin, OUTPUT);

pinMode(greenLedPin, OUTPUT);

Serial.begin(**9600**);

digitalWrite(redLedPin, HIGH);

digitalWrite(greenLedPin, LOW);

}

**void** **loop**() {

**int** sensorValue = analogRead(photoresistorPin);

Serial.println(sensorValue);

**if** (sensorValue > threshold) {

digitalWrite(greenLedPin, HIGH);

digitalWrite(redLedPin, LOW);

Serial.println("Дверь открыта.");

delay(openDuration);

sensorValue = analogRead(photoresistorPin);

**if** (sensorValue <= threshold) {

digitalWrite(greenLedPin, LOW);

digitalWrite(redLedPin, HIGH);

Serial.println("Дверь закрыта.");

}

} **else** {

digitalWrite(greenLedPin, LOW);

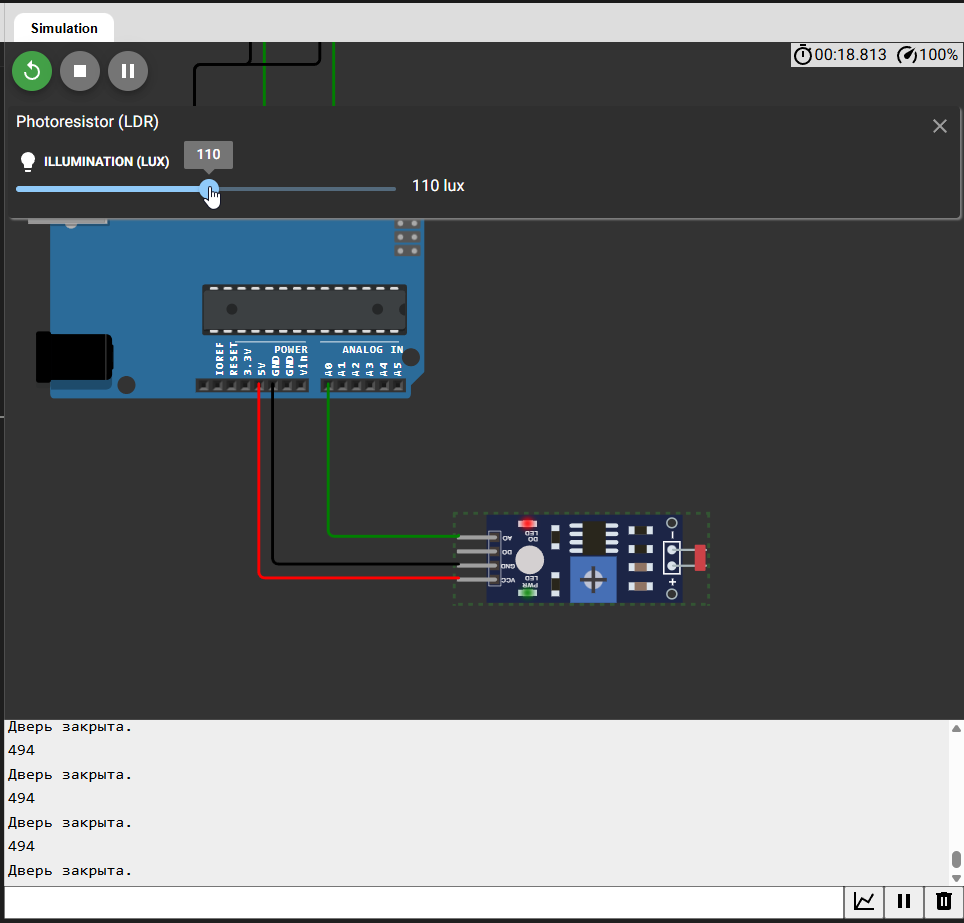
digitalWrite(redLedPin, HIGH);

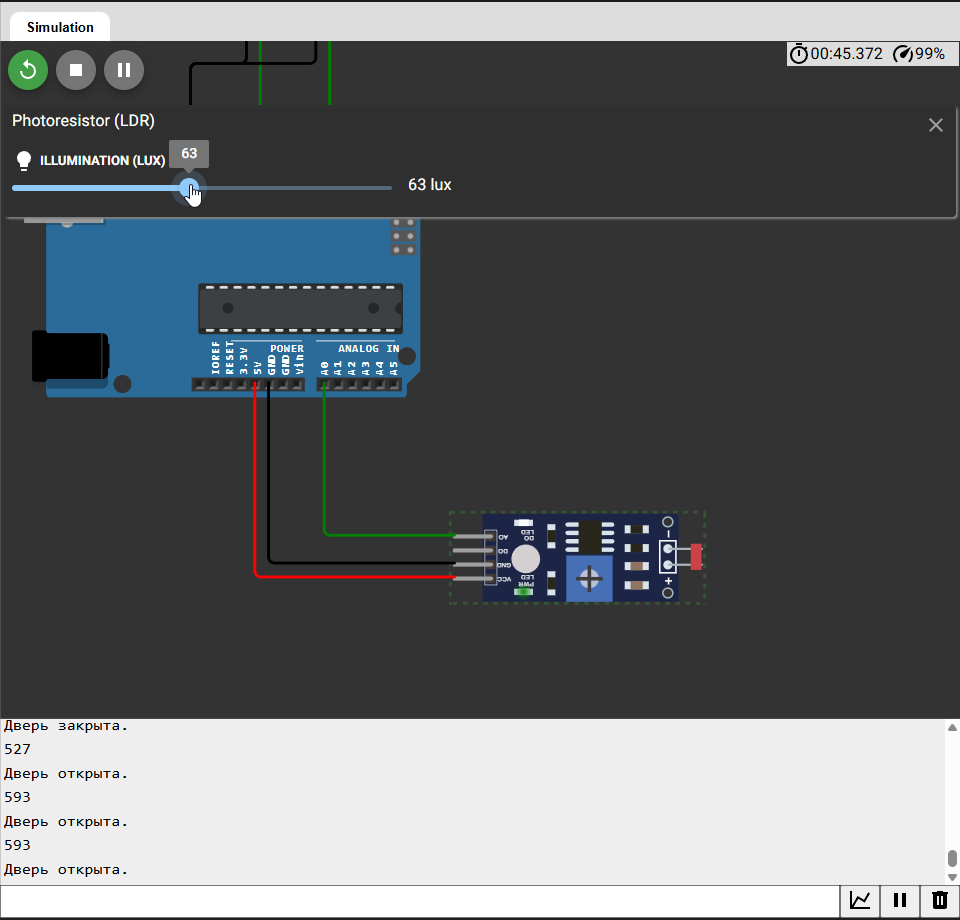
Serial.println("Дверь закрыта.");

}

}

Результат:

Рисунок 17 – пример выполнения программы 1

Рисунок 18 – пример выполнения программы 2

# **Задание №10**

Постановка задачи:

1. Собрать схему подключения сервопривода
2. Написать программу для управления сервоприводом через последовательный порт

Логика работы программы:

1. Программа находится в ожидании ввода данных в последовательный порт
2. Когда в последовательный порт вводятся данные, программа проверяет их на корректность (Должны приниматься только числовые значения в диапазоне от 0 до 180)
3. Если введён угол поворота, отличный от текущего угла, сервопривод плавно поворачивается в заданное положение.

Ход решения:

#include <Servo.h>

Servo myServo;

**int** currentAngle = **90**;

**int** targetAngle = **90**;

**const** **int** servoPin = **5**;

**void** **settings**() {

Serial.begin(**9600**);

myServo.attach(servoPin);

myServo.write(currentAngle);

Serial.println("Введите угол:");

}

**void** **setup**() {

settings();

}

**void** **loop**() {

**if** (Serial.available() > **0**) {

String input = Serial.readStringUntil('\n');

**int** angle = input.toInt();

**if** (angle >= **0** && angle <= **180**) {

targetAngle = angle;

moveServoSmoothly(currentAngle, targetAngle);

} **else** {

Serial.println("Ошибка: Угол должен быть до 180.");

}

}

}

**void** **moveServoSmoothly**(**int** startAngle, **int** endAngle) {

**int** step = (endAngle > startAngle) ? **1** : **-1**;

**for** (**int** angle = startAngle; angle != endAngle; angle += step) {

myServo.write(angle);

delay(**15**);

}

currentAngle = endAngle;

;

}

Результат:

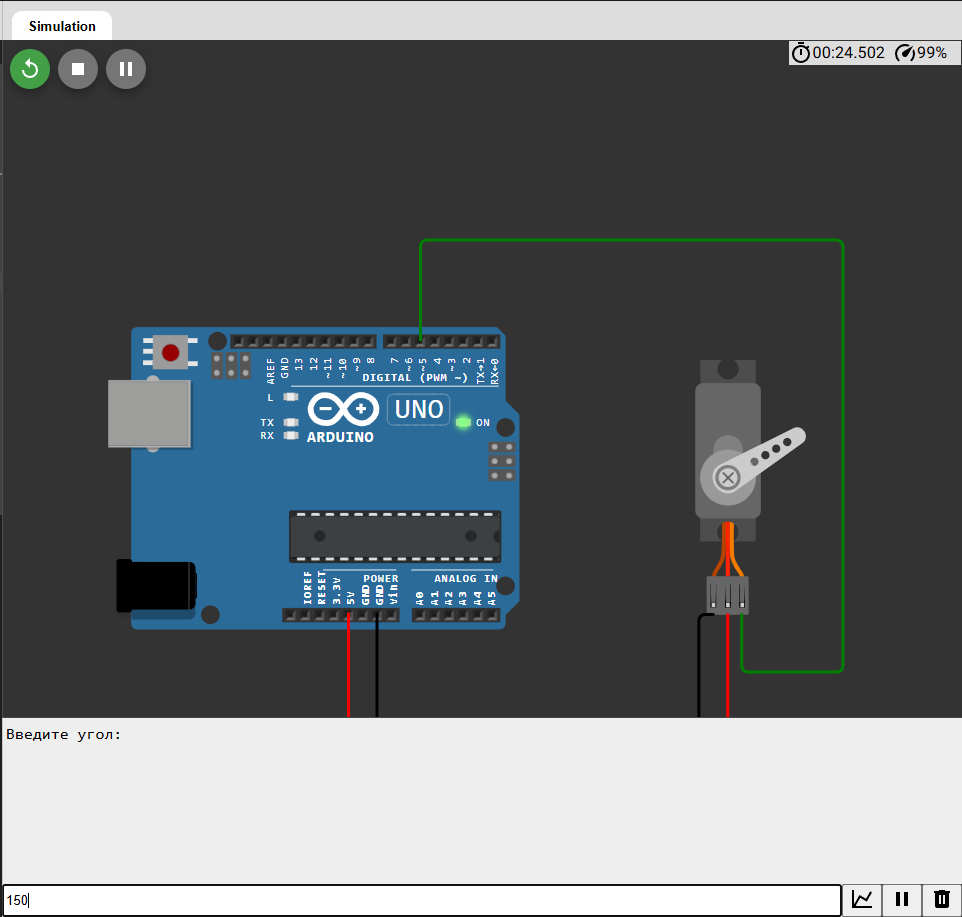


Рисунок 19 – пример выполнения программы 1

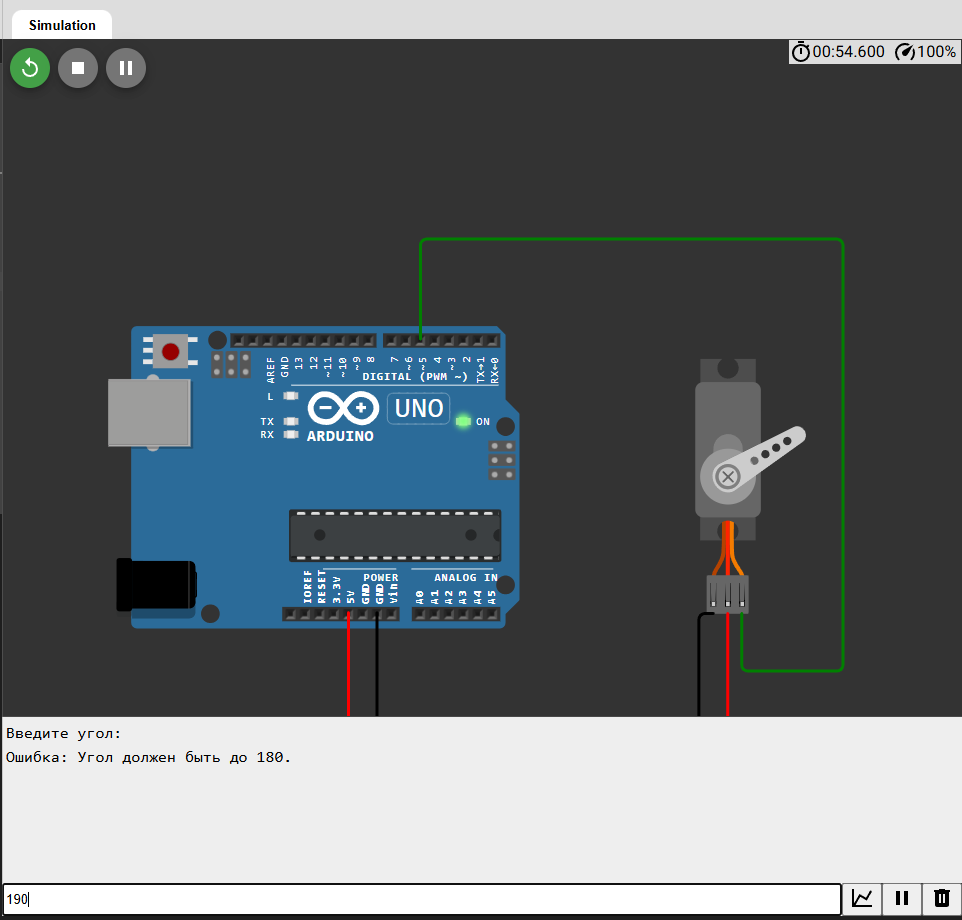


Рисунок 20 – пример выполнения программы 2

# **Задание №11**

Постановка задачи:

Дано изображение. На нем нужно найти все объекты желтого цвета и пометить их центры красной точкой.

Ход решения:

# 5 ВАРИАНТ

**import** **cv2**

**import** **numpy** **as** **np**

image = cv2.imread("test1.jpg")

cv2.imshow("original", image)

cv2.waitKey(**0**)

blurred\_image = cv2.GaussianBlur(image, (**11**, **11**), **0**)

cv2.imshow("blurred", blurred\_image)

cv2.waitKey(**0**)

hsv\_image = cv2.cvtColor(blurred\_image, cv2.COLOR\_BGR2HSV)

cv2.imshow("hsv", hsv\_image)

cv2.waitKey(**0**)

hsv\_min = np.array((**20**, **100**, **100**), np.uint8)

hsv\_max = np.array((**30**, **255**, **255**), np.uint8)

yellow\_mask = cv2.inRange(hsv\_image, hsv\_min, hsv\_max)

cv2.imshow("mask", yellow\_mask)

cv2.waitKey(**0**)

contours, hierarchy = cv2.findContours(yellow\_mask.copy(), cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

sorted\_contur = sorted(contours, key=cv2.contourArea, reverse=**True**)

**for** contour **in** sorted\_contur:

**if** cv2.contourArea(contour) > **100**:

x, y, w, h = cv2.boundingRect(contour)

print(x, y, w, h)

cv2.circle(image, (int(x + w / **2**), int(y + h / **2**)), **10**, (**0**, **0**, **255**), -**1**)

cv2.rectangle(image, (x, y), (int(x + w), int(y + h)), (**0**, **255**, **0**), **1**)

cv2.drawContours(image, [contour], -**1**, (**255**, **0**, **0**), **2**, cv2.LINE\_AA)

cv2.imshow('contours', image)

cv2.waitKey(**0**)

cv2.destroyAllWindows()

Результат:

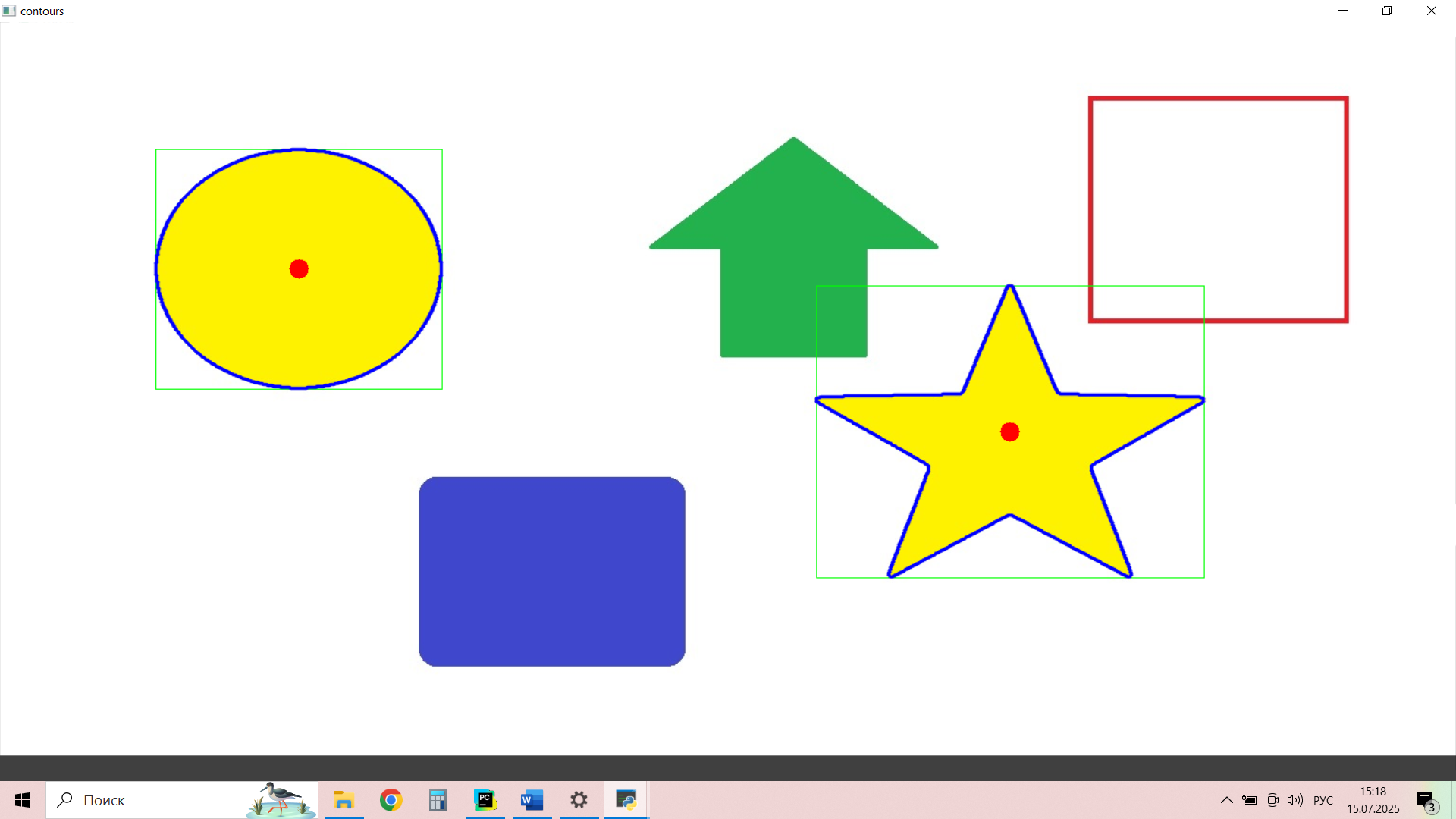


Рисунок 21 – пример выполнения программы 1

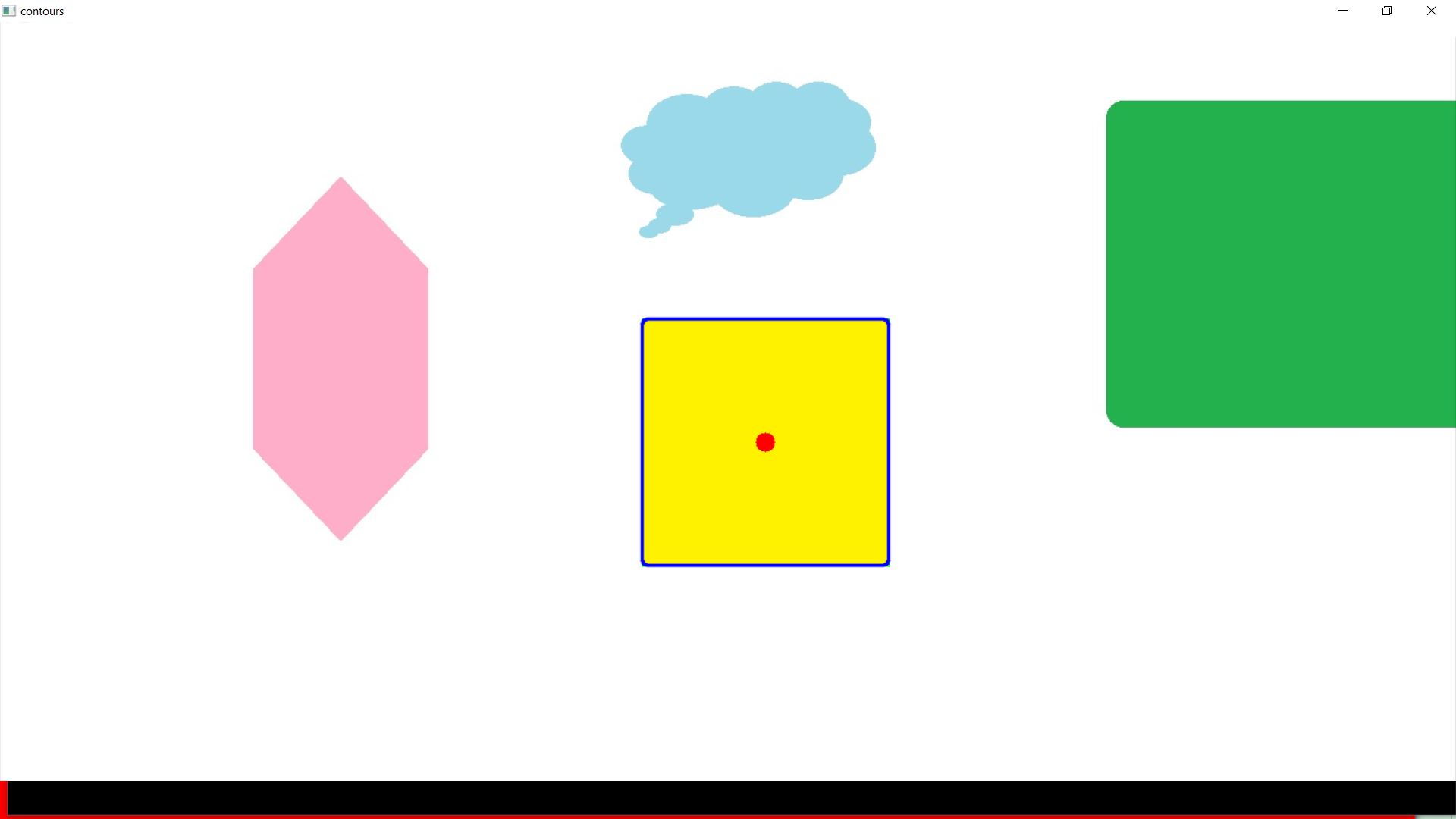


Рисунок 22 – пример выполнения программы 2

# **Вывод**

В период с 16 по 28 июня 2025 года я прошел учебную практику, целью которой являлось улучшение моих навыков как программиста и ознакомление с организациями, которые связаны с IT-сферой.

За это время было решено 11 задач. Первые 7 из них являлись прототипом тех задач, которые мы решали в течение первого курса, поэтому особых затруднений они не вызвали. Интереснее дела обстояли с оставшимися заданиями. В задачах под номерами 8, 9 и 10 нужно было написать программу для виртуального электронного устройства. 11-ая задача вообще была завязана на компьютерном зрении и машинном обучении. Так или иначе, изучать информацию для решения этих заданий было крайне интересно и увлекательно, и я думаю, что полученные знания в этих областях будут не лишними для меня в будущем.

Также еще стоит сказать, что в этот период наша группа посетила две экскурсии в двух разных организациях. Первое место, где нам посчастливилось побывать, стал филиал АО «СО ЕЭС» Иркутское РДУ. Данный филиал был создан в 2008 году. Он осуществляет функции оперативно-диспетчерского управления объектами электроэнергетики на территории Иркутской области. В первую очередь из запомнившегося хотел бы отметить многочисленность и современность серверов, потому как большую часть времени экскурсию проводили там, где эти и сервера располагались. Еще мне запомнился зал, в котором диспетчеры осматривают информацию об объектах на экране, состоящий по меньшей мере из 21 мониторов. Что-нибудь из рассказанного работниками организации мне сложно отметить, так как в основном повествование шло не совсем по теме IT. Но, подводя итог по этой экскурсии, не могу сказать, что она была бесполезной, и сам филиал на меня произвел большее впечатления, чем ожидал изначально.

Через день у нас была запланирована поездка в компанию «ISPsystem». Уже по названию можно было ожидать интересную экскурсию в организации, которая напрямую связана с IT-разработкой. Так оно и оказалось. «ISPsystem» — это российский разработчик платформ для комплексного управления ИТ-инфраструктурой. С 2004 года они создают софт для управления оборудованием, серверной виртуализацией, автоматизации учета и выдачи ресурсов. Когда мы прибыли туда, нас встретил офис компании, который выглядел как, в хорошем смысле этого слова, типичный офис, где работают IT-специалисты. Дальше нас провели и усадили в зал, где на протяжении двух часов коллектив из разработчиков рассказывали о компании, в которой они трудятся. Нам рассказали об истории «ISPsystem», о том, чем вообще занимается данная компания. Также за это время было рассеянно несколько мифов, на вроде того, что фронтенд-разработчик далеко не всегда на втором плане по сравнению с бэкенд-разработчиком. Презентация, посвященная тому, кем можно работать в компании, тоже была довольно информативна, ведь из нее можно было узнать о многообразии профессий, начиная с менеджера по продажам и заканчивая должностью, где человек наливает кофе. Самое важное для меня было сказано ближе к концу, когда речь зашла о собеседовании в «ISPsystem» и о прохождении производственной практике в этом месте. Для меня это было важно, потому что в такой компании, по моему впечатлению, приятно работать и что в этой компании можно работать, не покидая родной город. В общем и целом, поездка для меня оказалась крайне запоминающееся и, что самое главное, полезной, так как я планирую развиваться как разработчик.