

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 Ъ 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)



<https://irkutsk.hh.ru/resume/b7b8e3bcff0efccbde0039ed1f77567145514f>

Обучающегося Никитеевой В.Д., 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  
(ФИО, должность, подпись)

Оценка по практике \_\_\_\_\_  
(ФИО, подпись, дата)



<https://irkutsk.sup.erjob.ru/resume/razrabotchik-c-55731839.html>

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

Иркутск 2025

## Индивидуальное задание на прохождение

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

для \_\_\_\_\_ Никитеевой Виктории Дмитриевны  
(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:

Закрепление теоретических знаний, полученных в ходе изучения дисциплин «Информатика» и «Программирование», а также развитие практических навыков в области разработки программного обеспечения.

Содержание практики, вопросы, подлежащие изучению:

Совершенствование умений программирования на языке C++; работа с микроконтроллерной платформой Arduino и изучение принципов взаимодействия с различными модулями; освоение основ программирования на языке Python; знакомство с базовыми приёмами машинного зрения

Планируемые результаты практики:

Закрепление и углубление знаний по языку программирования C++; Приобретение навыков работы с Arduino; Формирование практических умений в области разработки программного обеспечения для управления устройствами; Получение первоначального опыта применения методов машинного зрения; Развитие навыков отладки, тестирования и документирования программного кода;

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(а), задание принято к исполнению

\_\_\_\_\_  
(подпись) «\_\_\_» \_\_\_\_\_ 2025 г.

## **ДНЕВНИК**

прохождения практики  
обучающегося Никитеевой Виктории Дмитриевны,  
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	Решила задачу №1, Решила задачу №2,	
2	17.06.2025	Составила резюме на hh.ru и superjob.ru, Решила задачу №3.	
3	18.06.2025	Решила задачу №4, Решила задачу №5.	
4	19.06.2025	Решила задачу №6, Решила задачу №7.	
5	20.06.2025	Изучила теоретический материал для задачи №8, Изучила теоретический материал для задачи №9.	
6	21.06.2025 – 23.06.2025	Решила задачу №8, Решила задачу №9. Изучила теоретический материал для задачи №10. Решила задачу №10.	
7	24.06.2025	Группой ходили в филиал АО Со Еэс Иркутское РДУ.	
8	25.06.2025	Изучила теоретический материал для задачи №11.	
9	26.06.2025	Группой ходили в IT-компанию ISPsystem.	
10	27.06.2025	Решила задачу №11.	

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

<b>Введение .....</b>	<b>6</b>
<b>Задание №1 .....</b>	<b>7</b>
<b>Задание №2 .....</b>	<b>10</b>
<b>Задание №3 .....</b>	<b>12</b>
<b>Задание №4 .....</b>	<b>15</b>
<b>Задание №5 .....</b>	<b>17</b>
<b>Задание №6 .....</b>	<b>19</b>
<b>Задание №7 .....</b>	<b>21</b>
<b>Задание №8 .....</b>	<b>22</b>
<b>Задание №9 .....</b>	<b>25</b>
<b>Задание №10 .....</b>	<b>28</b>
<b>Задание №11 .....</b>	<b>31</b>
<b>Отзыв о посещении филиала АО «СО ЕЭС» Иркутское РДУ .....</b>	<b>34</b>
<b>Отзыв о посещении компании ISPsystem .....</b>	<b>35</b>
<b>Заключение .....</b>	<b>36</b>
<b>Список литературы .....</b>	<b>37</b>

## **Введение**

Учебная практика на базе Федерального государственного бюджетного образовательного учреждения высшего образования «Иркутский национальный исследовательский технический университет» проходила в период с 16 по 28 июня 2025 года. Основная цель практики заключалась в закреплении знаний, полученных в ходе изучения профильных дисциплин, а также в развитии умений применять их на практике.

Особое внимание уделялось программированию на языке C++, формированию навыков анализа и решения задач, а также самостоятельному изучению нового материала. Для этого было выполнено одиннадцать заданий, каждое из которых направлено на развитие конкретных тем: от работы с алгоритмами и структурами данных до освоения инструментов, используемых в области информационных технологий.

Прохождение практики способствовало углублению профессиональных знаний и дало возможность применить теоретические основы в практической деятельности.

## Задание №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) *Генерирование горы (Generator)*: пользователь вводит высоту горы. Функция создает список строк, где каждая строка содержит на одно число больше предыдущей. Числа будем генерировать случайные (от 1 до 1000).
- 2) *Нахождение минимального пути (FindMinPath)*: используем динамическое программирование и будем идти снизу вверх, где  $dp[i][j]$  содержит минимальную сумму от вершины  $pyramid[i][j]$  до основания. На каждом шаге будем рассматривать минимальный из двух возможных путей вниз.
- 3) *Восстановление пути*: после вычислений восстанавливаем путь по значению  $dp$ , начиная от вершины и выбирая минимальный соседний элемент на каждом уровне.
- 4) *Вывод*: выводим последовательность вершин, соответствующую восстановленному минимальному пути, вместе с минимальной суммой времени.

### Код программы:

```
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>
#include <algorithm>

// Функция генерации горы со случайными числами в диапазоне [1, 1000].
std::vector<std::vector<int>> Generator(int size) {
    std::vector<std::vector<int>> mountain; // создаем гору.
    for (int i = 0; i < size; i++) {
        std::vector<int> row; // временный вектор, содержащий значения текущего уровня горы.
        for (int j = 0; j <= i; j++) {
            row.push_back(rand() % 1000 + 1); // генерация случайного числа.
        }
        mountain.push_back(row); // добавляем число в текущую строку row.
    }
    return mountain;
}
```

Рисунок 1 1 часть кода

```

// Поиск минимального пути.
std::pair<int, std::vector<int>> FindMinPath(const std::vector<std::vector<int>>& mountain) {
    int size = mountain.size(); // храним высоту горы.
    std::vector<std::vector<int>> dp = mountain; // создаем копию горы, в которую будем записывать минимальные суммы на каждом шаге.

    // Строим dp снизу вверх.
    for(int i = size - 2; i >= 0; --i) {
        for (int j = 0; j <= i; j++) {
            dp[i][j] += std::min(dp[i+1][j], dp[i+1][j+1]);
            // обновляем значения по самому минимальному соседу.
            // В dp[0][0] будет храниться минимальная сумма пути от вершины до основания.
        }
    }

    // Восстановление пути.
    std::vector<int> path;
    path.push_back(mountain[0][0]); // добавляем верхний элемент.
    int col = 0; // индекс, по которому отслеживаем, где сейчас находимся.
    for (int i = 1; i < size; i++) {
        if (dp[i][col] < dp[i][col+1]) { // смотрим соседей.
            path.push_back(mountain[i][col]); // идем влево вниз.
        } else {
            path.push_back(mountain[i][col+1]); // идем вправо вниз.
            col++; // сдвигаем индекс вправо.
        }
    }
    return {dp[0][0], path};
    // возвращаем пару значений, dp[0][0] — это минимальная сумма пути от вершины до основания,
    // path — последовательность чисел, по которым проходил минимальный путь.
}

```

Рисунок 2 2 часть кода

```

int main() {
    srand(time(0)); // инициализация генератора случайных чисел.

    int size = 0;
    std::cout << "Высота: ";
    std::cin >> size;

    auto mount = Generator(size); // генерация горы.

    // вывод горы.
    for (const auto& row : mount) {
        for (int num : row) {
            std::cout << num << " ";
        }
        std::cout << std::endl;
    }

    // поиск кратчайшего пути.
    auto result = FindMinPath(mount);
    int min = result.first;
    std::vector<int> path = result.second;

    // вывод результата.
    std::cout << "Минимальная сумма: " << min << std::endl;
    std::cout << "Путь: ";
    for (int num : path) {
        std::cout << num << " ";
    }
    std::cout << std::endl;
    return 0;
}

```

Рисунок 3 3 часть кода



**Таблица тестов:**

Ввод	4	5
Вывод	968 212 427 912 914 221 630 806 272 292 Минимальная сумма: 1888 Путь: 968 427 221 272	928 303 665 64 211 868 49 166 741 533 25 969 242 782 80 Минимальная сумма: 1369 Путь: 928 303 64 49 25

```
Высота: 10
58
468 285
898 342 29
613 136 120 222
618 209 581 179 88
656 566 23 598 321 222
280 192 478 407 845 997 561
699 70 116 811 215 443 138 188
450 296 279 956 71 231 435 801 54
918 323 202 426 50 816 244 695 804 570
Минимальная сумма: 1796
Путь: 58 285 342 136 209 23 407 215 71 50
```

Рисунок 4 Пример теста из задачи № 1

## Задание №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) *Представление сети в виде графа:* вершины графа соответствуют узлам сети, а рёбра — существующим линиям связи между ними.
- 2) *Поиск компонент связности:* с помощью обхода графа в глубину (DFS) программа определяет количество компонент связности в графе. Компонента связности — это множество узлов, которые можно достичь друг из друга, следуя по существующим линиям связи.
- 3) *Подсчёт дополнительных линий связи:* если в графе существует более одной компоненты связности, то необходимо построить дополнительные линии связи для объединения этих компонент. Минимальное количество таких линий равно количеству компонент минус один.
- 4) *Вывод результата:* программа выводит минимальное количество линий связи, которые нужно добавить, чтобы сделать граф связным, то есть чтобы все узлы были соединены в единую сеть.

### Код программы:

```
#include <iostream>
#include <vector>
#include <list>

// используем функцию DFS для обхода графа в глубину и поиска всех вершин.
void DFS(int v, const std::vector<std::list<int>>& adj, std::vector<bool>& visited) {
    visited[v] = true; // стартовая вершина изначально считается посещенной.
    for (int u : adj[v]) { // проходим по всем соседям вершины v.
        if (!visited[u]) { // если сосед еще не посещен, запускаем рекурсию.
            DFS(u, adj, visited);
        }
    }
}

// Функция для подсчета количества компонент связности.
int CountConnectedComponents(int N, const std::vector<std::pair<int, int>>& edges) {
    // Создаем список смежности.
    std::vector<std::list<int>> adj(N + 1); // Нумерация узлов с 1 до N.

    // Заполняем список смежности
```

Рисунок 5 Программа 2.1

```
// Заполняем список смежности
for (const auto& edge : edges) {
    int u = edge.first;
    int v = edge.second;
    adj[u].push_back(v);
    adj[v].push_back(u); // граф неориентированный.
}

std::vector<bool> visited(N + 1, false); // массив для отслеживания посещенных вершин.
int components = 0; // счетчик компонент.

// запускаем DFS из каждой непосещенной вершины.
for (int i = 1; i <= N; ++i) {
    if (!visited[i]) {
        DFS(i, adj, visited);
        components++; // увеличиваем количество компонент.
    }
}
return components;
}

int main() {
    int n = 0;
    int m = 0;
    std::cin >> n >> m;

    //заполнения списка ребер
    std::vector<std::pair<int, int>> edges;
    for (int i = 0; i < m; ++i) {
        int u = 0;
        int v = 0;
        std::cin >> u >> v;
        edges.emplace_back(u, v);
    }

    int components = CountConnectedComponents(n, edges);
    // Чтобы объединить все компоненты, нужно (components - 1) новых ребер
    std::cout << components - 1 << "\n";
    return 0;
}
```

Рисунок 6 Программа 2.2

Таблица тестов:

Ввод	10 6	7 4
	1 2	1 2
	2 8	2 3
	4 10	4 5
	5 9	6 7
	6 10	
	7 9	
Вывод	3	2

```
5 4
1 2
2 3
3 4
4 5
0
```

Рисунок 7 пример теста из задачи №2

### Задание №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.

1) Информация о снеге передается вам в виде тройки чисел – 1 в качестве идентификатора события, уникального индекса улицы и количество миллиметров выпавшего снега.

2) 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*: создадим структуру запросов.
- 2) *Обработка запросов*: Программа принимает  $m$  запросов двух типов:
  - Тип 1: обновляет количество снега на конкретной улице  $a$ . Значение  $u$  записывается в соответствующий индекс массива.
  - Тип 2: вычисляет и выводит сумму снега на отрезке улиц от  $a$  до  $b$  (включительно). Программа проходит по элементам массива в этом диапазоне и суммирует значения.
- 3) *Вывод результатов*: выводим результат суммирования по запросам типа 2.

#### Код программы:

```
#include <iostream>
#include <vector>

// Структура для хранения запроса.
struct Query {
    int type;
    int a;
    int b;
};
```

Рисунок 8 Программа 3.1

```

// Обработка всех запросов.
std::vector<int> process_queries(int n, const std::vector<Query>& queries) {
    std::vector<int> snow(n + 1, 0); // Вектор для хранения количества снега.
    std::vector<int> results; // Результаты для запросов типа 2.

    for (const Query& q : queries) {
        if (q.type == 1) {
            // Запрос: добавить снег на улице.
            snow[q.a] += q.b;
        } else if (q.type == 2) {
            // Запрос: посчитать сумму на отрезке.
            int sum = 0;
            for (int i = q.a; i <= q.b; ++i) {
                sum += snow[i];
            }
            results.push_back(sum); // Сохраняем результат.
        }
    }
    return results;
}

int main() {
    int n = 0;
    int k = 0;
    std::cin >> n >> k;

    std::vector<Query> queries;

    // Считываем все запросы
    for (int i = 0; i < k; ++i) {
        int type, a, b;
        std::cin >> type >> a >> b;
        queries.push_back({type, a, b});
    }

    // Обработываем запросы и получаем результаты
    std::vector<int> results = process_queries(n, queries);

    // Выводим результаты
    for (int res : results) {
        std::cout << res << '\n';
    }

    return 0;
}

```

Рисунок 9 Программа 3.2

### Таблица тестов:

Ввод	6 5 2 1 6 1 3 2 2 2 4 1 6 3 2 1 6	0 2 5
Вывод	5 3 1 3 7 1 1 4 2 1 5	11

```
5 4
1 2 3
1 4 7
2 1 5
2 2 4
10
10
```

Рисунок 10 пример теста из задачи №3

## Задание №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 *sinoptsnrtiora*. К этому слову снова применим эту же перестановку и получим *roartsnoitsin*. 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) Входные данные:

- Размер строки  $n$  и количество итераций  $k$ .
- Перестановка  $p$ .
- Зашифрованная строка.

#### 2) Описание перестановки: перестановка $P$ представляет собой набор чисел от 1 до $n$ , где каждое число соответствует новому порядку символов в слове. Например, если первая позиция перестановки содержит число 5, это означает, что первый символ новой строки будет взят с пятой позиции исходной строки.

#### 3) Процесс декодирования:

- Строим обратную перестановку: для каждой  $i$  позиции находим  $\text{invP}[p[i] - 1] = i + 1$ .
- Применяем  $\text{invP}$  к строке  $k$  раз (перемешиваем символы обратно).

#### 4) Результат: после применения перестановки $m$ раз, программа выводит восстановленное слово, которое является результатом декодирования зашифрованного слова.

### Код программы:

```
//  
#include <iostream>  
#include <vector>  
#include <string>  
  
// Функция для вычисления обратной перестановки.  
std::vector<int> Inverse(std::vector<int>& p) {  
    int n = p.size();  
    std::vector<int> invP(n); // вектор для хранения обратной перестановки.  
    for (int i = 0; i < n; i++) {  
        invP[p[i] - 1] = i + 1; // выполняем перестановку.  
    }  
    return invP;  
}
```

Рисунок 11 Программа 4.1

```

// Функция для применения перестановки p к строке s.
std::string Permutation(std::string& s, std::vector<int>& p) {
    int n = p.size();
    std::string result(n, ' '); // Создаём новую строку длины n, заполненную пробелами.
    for (int i = 0; i < n; i++) {
        // символ с позиции p[i] - 1 переносится на позицию i.
        result[i] = s[p[i] - 1];
    }
    return result;
}

int main() {
    int n = 0;
    int k = 0;
    std::cin >> n >> k;

    std::vector<int> p(n);
    for (int i = 0; i < n; i++) {
        std::cin >> p[i];
    }

    std::string world;
    std::cin >> world;

    std::vector<int> invp = Inverse(p); // вычисляем обратную перестановку.

    std::string result = world;
    for (int i = 0; i < k; i++) { // k раз применяем обратную перестановку.
        result = Permutation(result, invp);
    }

    std::cout << result << "\n";
    return 0;
}

```

Рисунок 12П Программа 4.2

### Таблица тестов:

Ввод	13 2 5 11 13 12 6 1 8 4 10 9 7 2 3 poartsnoitsin	5 1 2 3 4 5 1 bcdea
Вывод	transposition	abcde

```

4 100
1 2 3 4
abcd
abcd

```

Рисунок 13 пример теста из задачи №4



## Задание №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) Ввод данных:

- Считываем высоту  $h$  и ширину  $w$  матрицы.
- Вводим саму матрицу размером  $h \times w$ , содержащую 0 и 1.

#### 2) Поиск границ фигуры:

- Инициализируем переменные:
  - ❖  $\text{min\_row} = \text{INT\_MAX}$  — минимальный индекс строки, где есть 1;
  - ❖  $\text{max\_row} = \text{INT\_MIN}$  — максимальный индекс строки, где есть 1;
  - ❖  $\text{min\_col} = \text{INT\_MAX}$  — минимальный индекс столбца, где есть 1;
  - ❖  $\text{max\_col} = \text{INT\_MIN}$  — максимальный индекс столбца, где есть 1.
- Проходим по каждому элементу матрицы. Если элемент равен 1, обновляем границы:
  - ❖  $\text{min\_row} = \min(\text{min\_row}, i)$ ;
  - ❖  $\text{max\_row} = \max(\text{max\_row}, i)$ ;
  - ❖  $\text{min\_col} = \min(\text{min\_col}, j)$ ;
  - ❖  $\text{max\_col} = \max(\text{max\_col}, j)$ .

#### 3) Расчёт и вывод координат:

- Для того чтобы фигура не касалась сторон прямоугольника, уменьшаем на 1 значения  $\text{min\_row}$  и  $\text{min\_col}$ , а значения  $\text{max\_row}$  и  $\text{max\_col}$  увеличиваем на 1.
- Выводим координаты верхнего левого угла ( $\text{min\_row} - 1, \text{min\_col} - 1$ ) и нижнего правого угла ( $\text{max\_row} + 1, \text{max\_col} + 1$ ).

### Код программы:

```
#include <iostream>
#include <vector>
#include <climits>

int main() {
    int h = 0;
    int w = 0;
    std::cin >> h >> w;
```

Рисунок 14 Программа 5.1

```

std::vector<std::vector<int>> matrix(h, std::vector<int>(w));
// Считываем матрицу
for (int i = 0; i < h; ++i) {
    for (int j = 0; j < w; ++j) {
        std::cin >> matrix[i][j];
    }
}

// Начальные значения для поиска минимума и максимума
int minRow = INT_MAX, maxRow = INT_MIN;
int minCol = INT_MAX, maxCol = INT_MIN;

// Проходим по всей матрице
for (int i = 0; i < h; ++i) {
    for (int j = 0; j < w; ++j) {
        if (matrix[i][j] == 1) {
            // Обновляем границы прямоугольника
            if (i < minRow) minRow = i;
            if (i > maxRow) maxRow = i;
            if (j < minCol) minCol = j;
            if (j > maxCol) maxCol = j;
        }
    }
}

// Выводим координаты: верхний левый угол и нижний правый угол
std::cout << minRow - 1 << " " << minCol - 1 << " " << maxRow + 1 << " " << maxCol + 1 << std::endl;

return 0;
}

```

Рисунок 15 Программа 5.2

### Таблица тестов:

Ввод	10 10 1 1 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 1 1 0	10 10 1 0
Вывод	3 2 8 6	3 3 5 5

## Задание №6

В школьном кружке робототехники есть два вида микроконтроллеров (условно тип А и тип В) и два вида модулей управления мотором (условно тип 1 и тип 2). Выяснилось, что контроллер типа В и модуль управления типа 2 несовместимы. Использование микроконтроллеров и модулей управления в других комбинациях возможно. Имеется а микроконтроллеров типа А, b микроконтроллеров типа В, 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) Так как, контроллеры типа В не совместимы с модулями типа 2, то мы можем совместить их только с 1 типом модулей управления.
- 2) Оставшиеся модули типа 1 и все модули типа 2 можно использовать с контроллерами типа А.
- 3) Тогда общее количество пар – это сумма пар котроллер типа В + модуль 1 и контроллер типа А + модуль 1 или 2.

### Код программы:

```
#include <iostream>
#include <vector>
#include <algorithm>

int main() {
    int n = 0; // количество строк.
    std::cin >> n;
    std::vector<int> otvet; // вектор для ответа, подсчитанного количества пар.
    for (int i = 0; i < n; i++) {
        int a, b, x, y; // объявление переменных.
        std::cin >> a >> b >> x >> y;

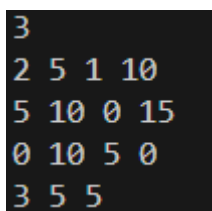
        int min_bx = std::min(b, x); // контроллеры В используют модули типа 1.
        int diff = x - min_bx; // Оставшиеся модули типа 1.
        int used_a = std::min(a, diff + y); // контроллеры А для оставшихся модулей.

        int result = min_bx + used_a;
        otvet.push_back(result);
    }
    for (int var : otvet) {
        std::cout << var << " ";
    }
    return 0;
}
```

Рисунок 16 Программа 6

**Таблица тестов:**

Ввод	3 5 2 3 7 10 2 12 2 3 8 5 6	1 100 100 100 100
Вывод	7 12 8	200



```
3
2 5 1 10
5 10 0 15
0 10 5 0
3 5 5
```

Рисунок 17 пример теста из задачи №6

## Задание №7

На компьютере работника автосервиса нашли файл с последовательностью автомобильных номеров, обслуживающихся 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:

Проверяем каждую букву (цифру) находится ли она на своём месте.

### Код программы:

```
int main() {
#include <iostream>
#include <string>
#include <cctype>

std::string str;
std::cin >> str;

bool is_valid = true;

if (str.length() != 6) {
    is_valid = false;
}
else if (!isupper(str[0])) {
    //isupper() проверяет является ли символ прописным.
    is_valid = false;
}
else if (!isdigit(str[1]) || !isdigit(str[2]) || !isdigit(str[3])) {
    //isdigit() проверяет является ли символ цифрой.
    is_valid = false;
}
else if (!isupper(str[4]) || !isupper(str[5])) {
    is_valid = false;
}
std::cout << (is_valid ? "Yes" : "No") << "\n";
return 0;
}
```

Рисунок 18 Программа 7

### Таблица тестов:

Ввод	K040LE	M3239L
Вывод	Yes	No

## Задание №8

Составить светодиодную матрицу размером не менее 8 на 8 светодиодов. На матрицу вывести инфографику с различными динамично меняющимися изображениями.

### Алгоритм программы:

- 1) *Инициализация:*
  - Подключаем библиотеку Adafruit\_NeoPixel, которая позволяет управлять адресными светодиодами.
  - Устанавливаем параметры матрицы: количество светодиодов (64), пин подключения, яркость, ширина и высота матрицы.
  - Инициализируем объект ленты, производим запуск и установку яркости.
- 2) *Формирование координатной системы:*
  - Создаем функции getPixelIndex(x, y) и setPixelXY(x, y, color) для перевода координат матрицы в индекс ленты и установки цвета по координатам. Учитываем змеиный порядок подключения (чётные строки слева направо, нечётные — справа налево).
- 3) *Отображение эффектов:*
  - В главном цикле loop() поочередно запускаем два эффекта:
    - ❖ Сердце — статичное изображение сердца на 8×8 матрице.
    - ❖ Радуга — динамичный эффект переливания всех цветов радуги.
- 4) *Рисование на матрице:*
  - В каждом эффекте используется логика покадровой отрисовки:
    - ❖ Сердце формируется из двумерного массива и рисуется как фиксированное изображение.
    - ❖ Радуга создаётся циклическим изменением оттенка каждого пикселя на основе HSV-модели.

### Код программы:

```
#include <Adafruit_NeoPixel.h>

// Параметры подключения и матрицы
#define LED_PIN    6      // Пин Arduino, к которому подключена лента WS2812
#define LED_COUNT  64     // Общее количество светодиодов (8x8)
#define BRIGHTNESS 100    // Яркость (0-255)
#define WIDTH      8      // Ширина матрицы
#define HEIGHT     8      // Высота матрицы

// Создание объекта ленты
Adafruit_NeoPixel strip(LED_COUNT, LED_PIN, NEO_GRB + NEO_KHZ800);
```

Рисунок 19 Программа 8.1

```

// Функция для перевода координат (x, y) в индекс светодиода
int getPixelIndex(int x, int y) {
    // Змейка: чётные строки слева направо, нечётные — справа налево
    if (y % 2 == 0) {
        return y * WIDTH + x;
    } else {
        return y * WIDTH + (WIDTH - 1 - x);
    }
}

// Установка цвета пикселя по координатам x, y
void setPixelXY(int x, int y, uint32_t color) {
    // Проверка, чтобы не выйти за границы массива
    if (x >= 0 && x < WIDTH && y >= 0 && y < HEIGHT) {
        strip.setPixelColor(getPixelIndex(x, y), color);
    }
}

void setup() {
    strip.begin();           // Инициализация ленты
    strip.show();            // Очистка (все пиксели выключены)
    strip.setBrightness(BRIGHTNESS); // Установка яркости
}

void loop() {
    showHeart(3);           // Показать сердце в течение 3 секунд
    rainbowEffect(20);      // Радужный эффект
}

```

Рисунок 20 Программа 8.2

```

// Вывод пиксельного изображения сердца
void showHeart(unsigned long duration) {
    // 8×8 пикселей: 1 — красный пиксель, 0 — выключен
    uint8_t heart[8][8] = {
        {0,1,1,0,0,1,1,0},
        {1,1,1,1,1,1,1,1},
        {1,1,1,1,1,1,1,1},
        {1,1,1,1,1,1,1,1},
        {0,1,1,1,1,1,0},
        {0,0,1,1,1,1,0,0},
        {0,0,0,1,1,0,0,0},
        {0,0,0,0,0,0,0,0}
    };

    unsigned long start = millis(); // Засекаем начало

    // Цикл, пока не пройдёт указанное время
    while (millis() - start < duration * 1000) {
        // Проходим по каждой точке и рисуем сердце
        for (int y = 0; y < 8; y++) {
            for (int x = 0; x < 8; x++) {
                // Если в массиве 1 — включаем пиксель красным, иначе — чёрный
                setPixelXY(x, y, heart[y][x] ? strip.Color(255, 0, 0) : 0);
            }
        }
        strip.show(); // Обновляем экран
        delay(50);
    }
}

```

Рисунок 21 Программа 8.3

```

// Радужная волна по всей матрице
void rainbowEffect(int wait) {
  for (long firstPixelHue = 0; firstPixelHue < 5 * 65536; firstPixelHue += 256) {
    for (int i = 0; i < strip.numPixels(); i++) {
      // Смещаем цвет каждого пикселя относительно начала
      int pixelHue = firstPixelHue + (i * 65536L / strip.numPixels());
      strip.setPixelColor(i, strip.gamma32(strip.ColorHSV(pixelHue))); // Преобразуем в RGB
    }
    strip.show();
    delay(wait);
  }
}

```

Рисунок 22 Программа 8.4

## Пример работы программы:

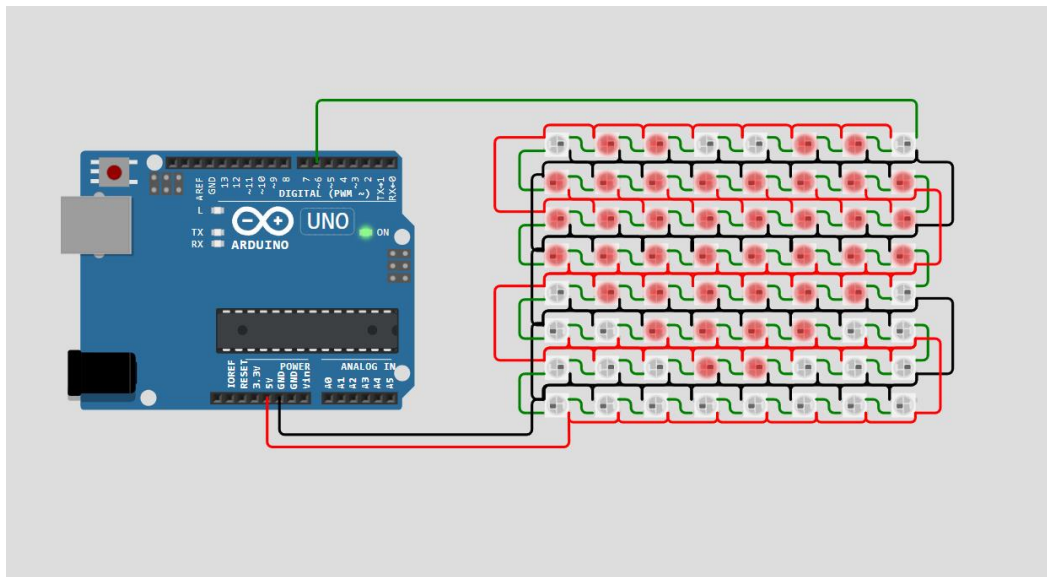


Рисунок 23 Изображение сердца

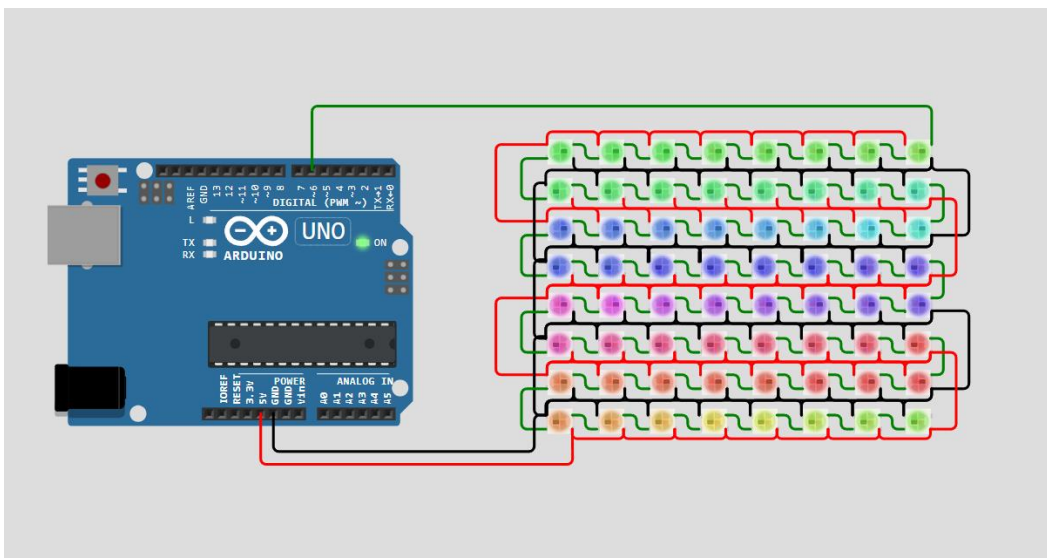


Рисунок 24 Изображение радуги



## Задание №9

### Задачи:

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

Зеленый светодиод – двери открываются.

Красный светодиод – двери закрываются.

Фоторезистор имитируют процесс приближения-удаления человека от дверей.

### Алгоритм программы:

Программа управляет светодиодами для имитации автоматических дверей: если фоторезистор фиксирует низкую освещённость (человек рядом), включается зеленый светодиод (двери открываются); если освещённость высокая (человек ушёл), включается красный светодиод (двери закрываются).

### Код программы:

```
#define greenLedPin 9      // зелёный светодиод – двери открыты
#define redLedPin 5        // красный светодиод – двери закрыты
#define sensorPin A0       // фоторезистор

// Время, на которое двери останутся открытыми.
const unsigned long openDuration = 3000;

void setup() {
  pinMode(redLedPin, OUTPUT);
  pinMode(greenLedPin, OUTPUT);
  Serial.begin(9600);

  // По умолчанию двери закрыты (горит красный)
  digitalWrite(redLedPin, HIGH);
  digitalWrite(greenLedPin, LOW);
}

void loop() {
  int sensorValue = analogRead(sensorPin);
  Serial.print("Значение с фоторезистора: ");
  Serial.println(sensorValue);

  // Если освещённость низкая (человек рядом) – открыть дверь
  if (sensorValue < 512) {
    // открыть двери (зелёный)
    digitalWrite(redLedPin, LOW);
    digitalWrite(greenLedPin, HIGH);
    Serial.println("Человек подошёл – двери открываются");

    delay(openDuration); // подождать
```

Рисунок 25 Программа 9.1

```

sensorValue = analogRead(sensorPin); // проверить снова
if (sensorValue < 512) {
  Serial.println("Человек всё ещё рядом — двери остаются открытыми");
} else {
  // закрыть двери (красный)
  digitalWrite(redLedPin, HIGH);
  digitalWrite(greenLedPin, LOW);
  Serial.println("Человек ушёл — двери закрываются");
}
} else {
  // человек далеко — двери закрыты
  digitalWrite(redLedPin, HIGH);
  digitalWrite(greenLedPin, LOW);
  Serial.println("Человек далеко — двери закрыты");
}

delay(500);
}

```

Рисунок 26 Программа 9.2

## Пример работы программы:

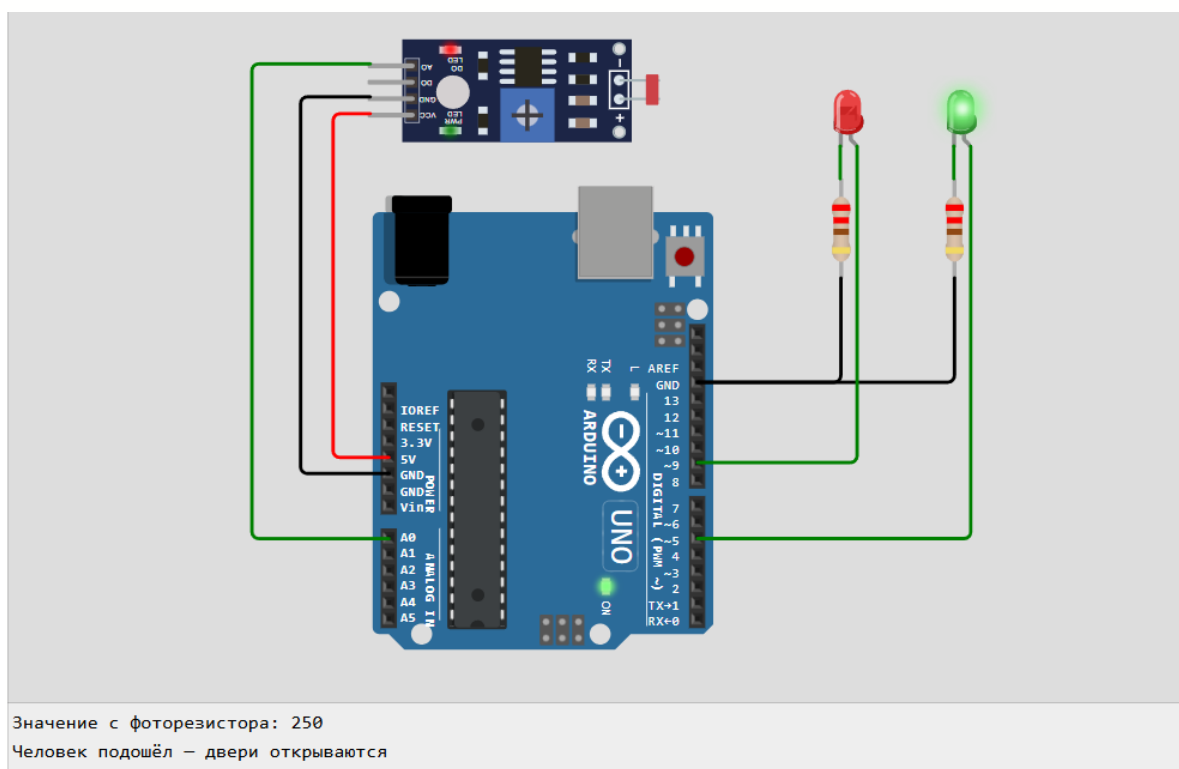


Рисунок 27 зеленый светодиод

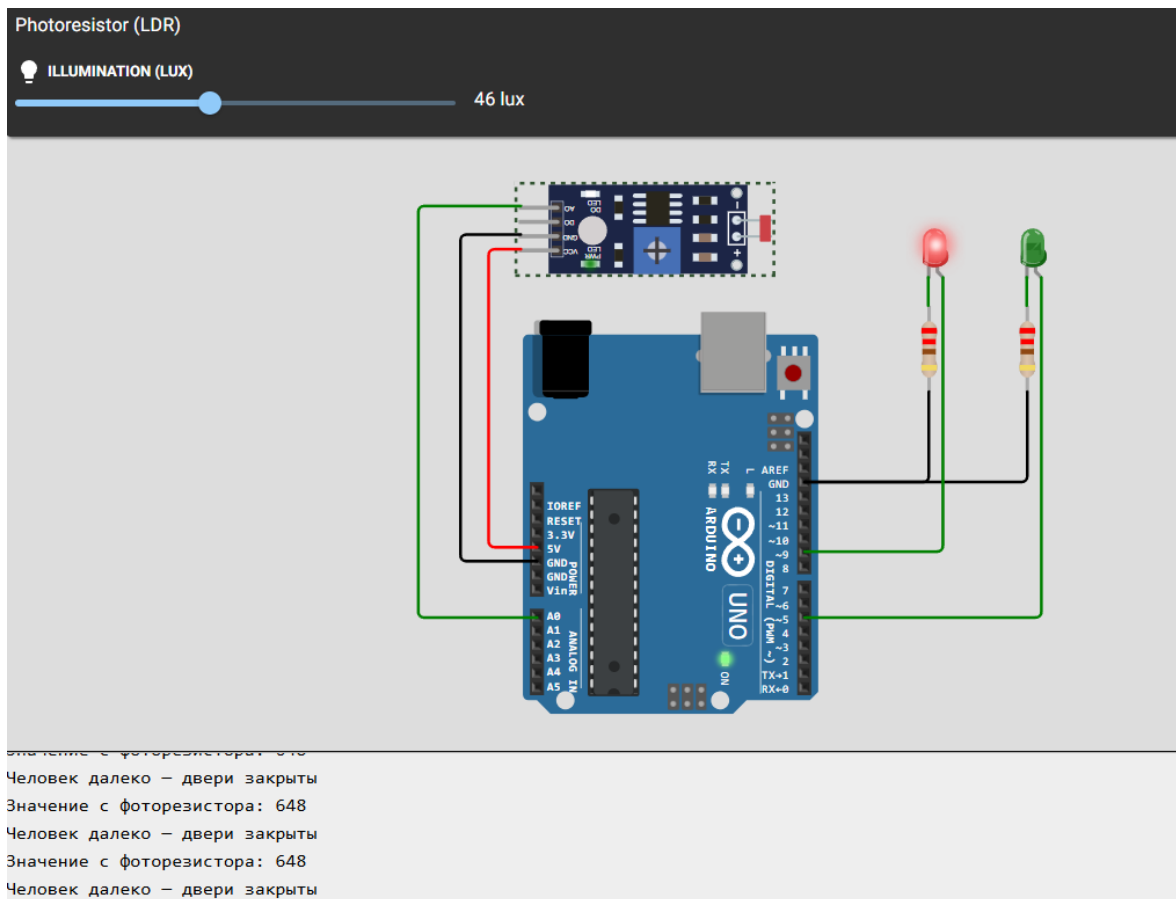


Рисунок 28 Красный светодиод

## Задание №10

### Задачи:

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

Логика работы программы: запрашивается угол поворота сервопривода, если он отличен от того, на который повернут привод, то плавно повернуть до указанного. Программа работает в цикле, с возможностью постоянно изменять угол поворота.

### Алгоритм программы:

- 1) *Инициализация:* программа настраивает связь с сервоприводом, устанавливая его начальное положение на угол 90 градусов. Также начинается обмен данными через последовательный порт для взаимодействия с пользователем.
- 2) *Чтение угла:* в цикле программа постоянно проверяет наличие данных в порту. Если введено новое значение угла (от 0 до 180 градусов), и оно отличается от текущего угла сервопривода, привод плавно поворачивается на указанный угол.
- 3) *Плавный поворот:* Программа сравнивает новый угол с текущим и постепенно изменяет угол сервопривода, чтобы обеспечить плавный переход.
- 4) *Повторение:* Программа работает в непрерывном цикле, позволяя пользователю изменять угол поворота сервопривода в любой момент.

### Код программы:

```
#include <Servo.h>

Servo myServo;
int currentAngle = 90; // Стартовое положение
int targetAngle = 90;

void setup() {
  Serial.begin(9600);      // Запуск последовательного порта
  myServo.attach(3);       // Подключение сервопривода к пину
  myServo.write(currentAngle); // Установка стартовой позиции
  Serial.println("Введите угол от 0 до 180:");
}
```

Рисунок 29 Программа 10.1

```

void loop() {
  if (Serial.available()) {
    String input = Serial.readStringUntil('\n'); // Считывание строки
    input.trim(); // Удаление пробелов и перевода строки

    if (isNumeric(input)) {
      int angle = input.toInt();

      if (angle >= 0 && angle <= 180) {
        if (angle != currentAngle) {
          moveServoSmooth(currentAngle, angle);
          currentAngle = angle;
        } else {
          Serial.println("Сервопривод уже в этом положении.");
        }
      } else {
        Serial.println("Ошибка: введите число от 0 до 180.");
      }
    } else {
      Serial.println("Ошибка: введите числовое значение.");
    }
  }
}

bool isNumeric(String str) {
  for (byte i = 0; i < str.length(); i++) {
    if (!isDigit(str.charAt(i))) return false;
  }
  return true;
}

void moveServoSmooth(int from, int to) {
  int step = (to > from) ? 1 : -1;
  for (int pos = from; pos != to; pos += step) {
    myServo.write(pos);
    delay(10);
  }
  myServo.write(to);
}

```

Рисунок 30 Программа 10.2

## Пример работы программы:

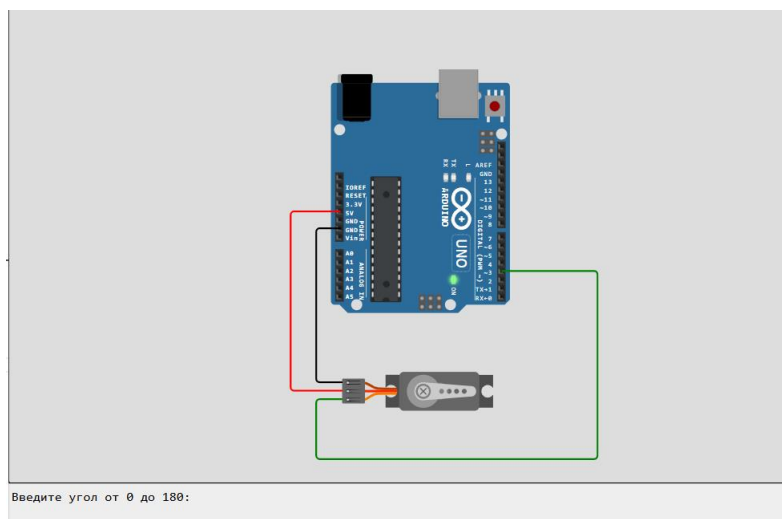


Рисунок 31 Начальное положение (90 градусов)

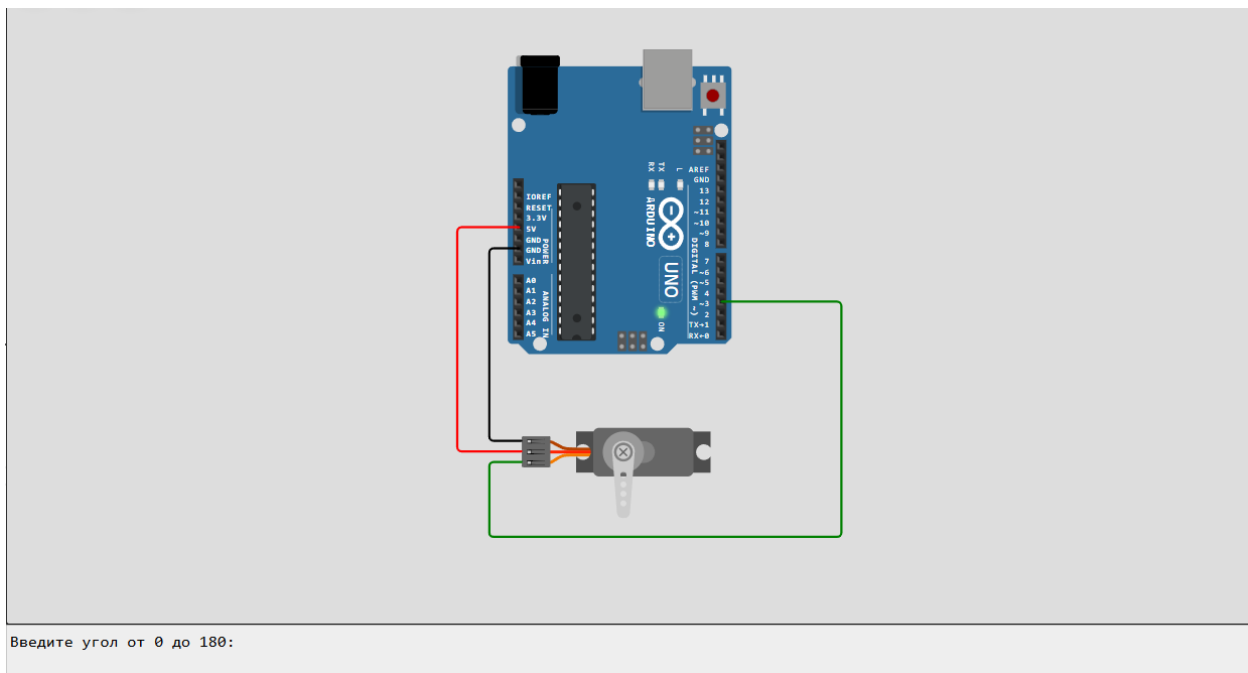


Рисунок 32 Поворот на 180 градусов

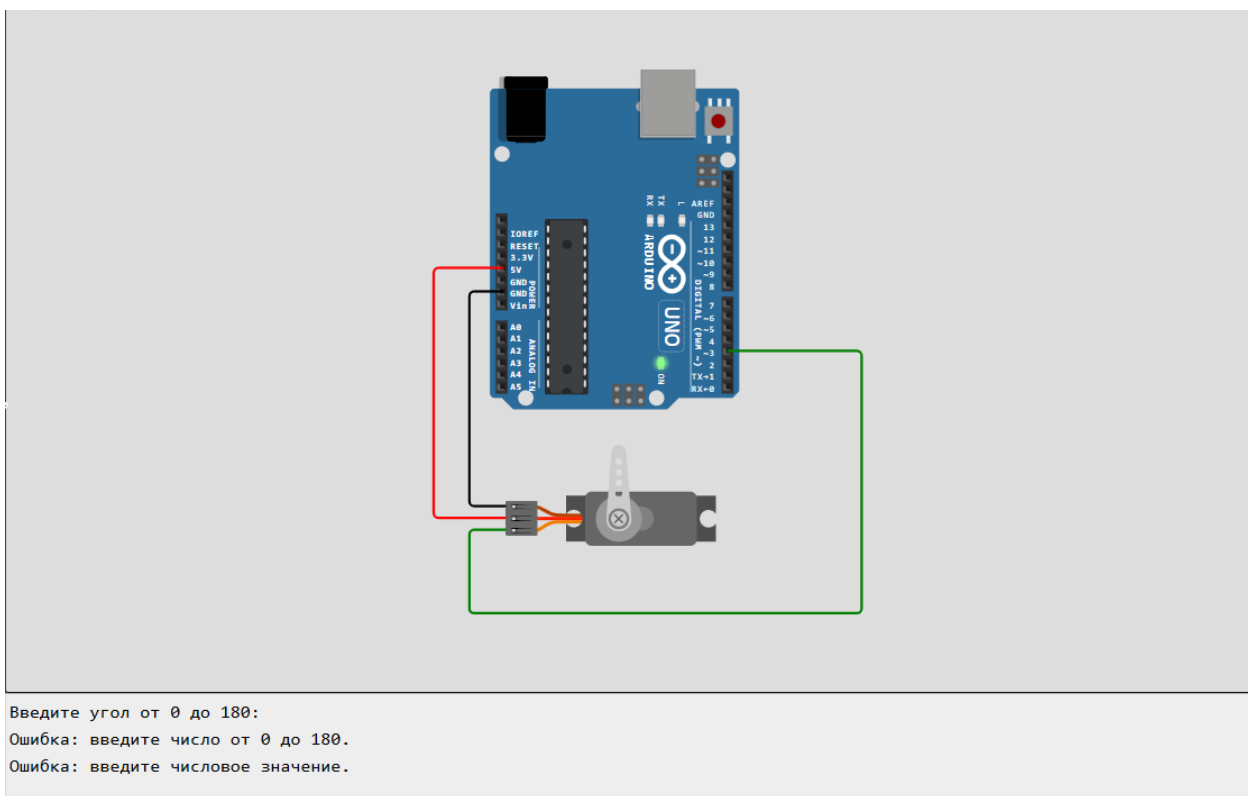


Рисунок 33 Положение на 0 и проверка на диапазон и число

## Задание №11

Find all green and yellow objects in the image. Find the centers of all green and yellow objects. Highlight the boundaries of green objects using a blue frame. Select the boundaries of yellow objects using a green frame.

Перевод:

Найдите все зеленые и желтые объекты на изображении. Найдите центры всех зеленых и желтых объектов. Выделите границы зеленых объектов синей рамкой. Выделите границы желтых объектов зеленой рамкой.

### Алгоритм программы:

- 1) *Преобразование изображения в HSV*: Изображение конвертируется в цветное пространство HSV для удобства работы с цветами.
- 2) *Обнаружение зеленых объектов*:
  - ❖ Задаются нижняя и верхняя границы зелёного цвета в HSV.
  - ❖ Создаётся маска, выделяющая все пиксели изображения, попадающие в диапазон зелёного.
  - ❖ На основе маски находятся контуры зелёных объектов.
  - ❖ Для каждого найденного контура:
    - ✓ Вычисляется центр.
    - ✓ Центр отмечается кружком, контур выделяется синим цветом.
- 3) *Обнаружение желтых объектов*: аналогично, но контур выделяется зеленым цветом.
- 4) *Отображение результата*: Обработанное изображение с выделенными зелёными и жёлтыми объектами и отмеченными центрами выводится на экран.

### Код программы:

```
import cv2
import numpy as np

# Загружаем изображение
image = cv2.imread("C:/Users/vikusik/Desktop/proga/praktika.cpp/test_2.jpg")

# Размытие для снижения шумов
blurred = cv2.GaussianBlur(image, (11, 11), 0)

# Перевод в HSV
hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)
```

Рисунок 34 Программа 11.1

```

# Зелёный цвет
green_min = np.array((36, 25, 25), np.uint8)
green_max = np.array((70, 255, 255), np.uint8)
green_mask = cv2.inRange(hsv, green_min, green_max)

# Поиск контуров зелёных объектов с иерархией
green_contours, green_hierarchy = cv2.findContours(green_mask.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

# Обрабатываем только внешние контуры (Parent == -1)
for i, contour in enumerate(green_contours):
    # Проверяем иерархию
    if green_hierarchy[0][i][3] == -1: # Parent == -1 значит внешний контур
        x, y, w, h = cv2.boundingRect(contour)
        center = (int(x + w/2), int(y + h/2))

        cv2.circle(image, center, 7, (0, 0, 255), 2)
        cv2.drawContours(image, [contour], -1, (255, 0, 0), 4)
        #cv2.rectangle(image, (x, y), (x + w, y + h), (255, 0, 0), 2)

# Жёлтый цвет
yellow_min = np.array((20, 100, 100), np.uint8)
yellow_max = np.array((36, 255, 255), np.uint8)
yellow_mask = cv2.inRange(hsv, yellow_min, yellow_max)

# Поиск контуров жёлтых объектов с иерархией
yellow_contours, yellow_hierarchy = cv2.findContours(yellow_mask.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

# Обрабатываем только внешние контуры
for i, contour in enumerate(yellow_contours):
    if yellow_hierarchy[0][i][3] == -1: # внешний контур
        x, y, w, h = cv2.boundingRect(contour)
        center = (int(x + w/2), int(y + h/2))

        cv2.circle(image, center, 7, (0, 0, 255), 2)
        cv2.drawContours(image, [contour], -1, (0, 255, 0), 4)
        #cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)

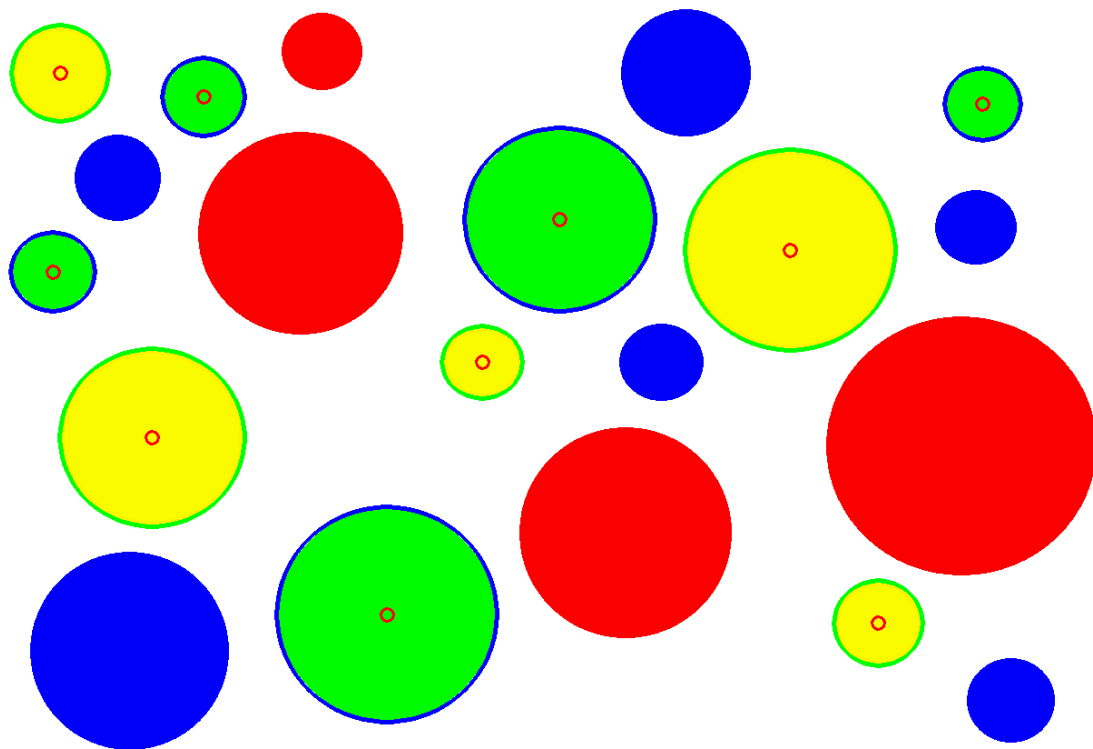
# Показ результата
cv2.imshow("result", image)
cv2.waitKey(0)
cv2.destroyAllWindows()

```

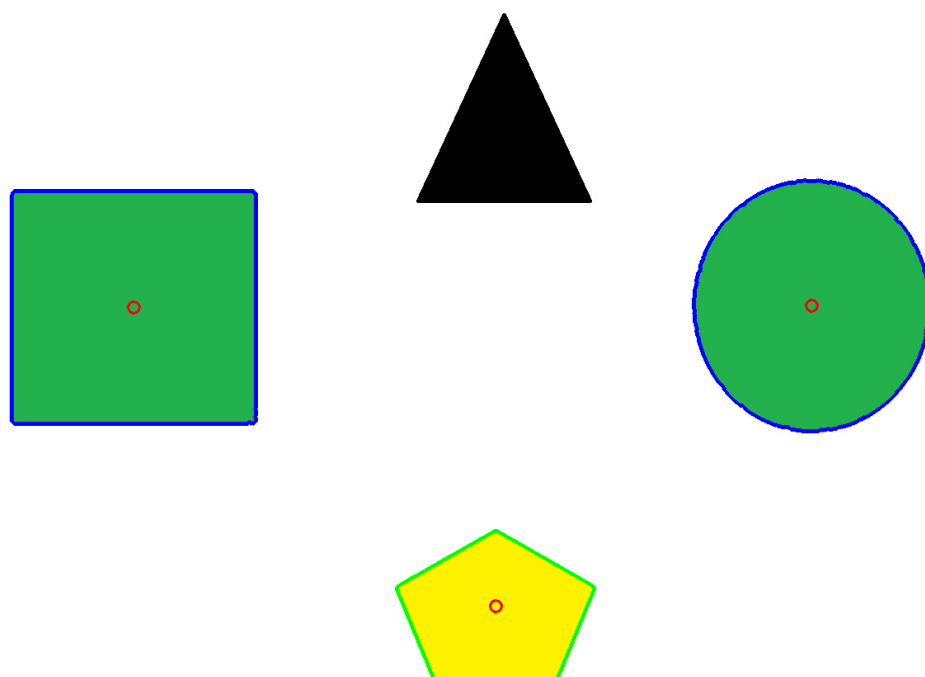
Рисунок 35 Программа 11.2



**Пример работы программы:**



*Рисунок 36 пример работы программы*



*Рисунок 37 пример работы программы*

## **Отзыв о посещении филиала АО «СО ЕЭС» Иркутское РДУ**

В рамках практики наша группа посетила филиал АО «Системный оператор Единой энергетической системы» — Иркутское региональное диспетчерское управление (РДУ). Эта организация играет ключевую роль в обеспечении надежного функционирования электроэнергетики Иркутской области. Основная деятельность компании связана с управлением энергосистемой региона, координацией работы электростанций и сетей, а также оперативным регулированием процессов производства и распределения электроэнергии.

На встрече было подчеркнуто, что с развитием отрасли всё большее значение приобретают информационные технологии. В компании функционирует специализированный ИТ-отдел, который разрабатывает и внедряет автоматизированные системы, обеспечивающие поддержку диспетчерского управления. Среди них можно выделить автоматизированный информационный комплекс (АИК), используемый для проведения расчетов и предоставления данных диспетчерам. Также внедряется современная симуляционная модель («информационный двойник»), позволяющая прогнозировать поведение электростанций и систем в различных режимах.

Посещение Иркутского РДУ позволило нам лучше понять особенности организации диспетчерского управления, важность информационных технологий в сфере энергетики, а также масштабы ответственности, которые несет компания за стабильность энергоснабжения региона.

## **Отзыв о посещении компании ISPsystem**

В рамках практики мы посетили компанию ISPsystem — разработчика программного обеспечения для управления IT-инфраструктурой. Важно отметить, что компания не занимается хостингом, заказной разработкой или поддержкой сторонних решений, а сосредоточена на создании и развитии собственных продуктов.

Сотрудники компании произвели очень приятное впечатление: они были доброжелательными и с увлечением рассказывали о своей работе. Было видно, что люди действительно гордятся тем, чем занимаются.

Нам также рассказали о направлениях, в которых требуются специалисты. Среди них: разработка (Backend, Frontend, QA, PM, UX/UI, аналитика), коммерция (маркетинг, продажи), техническая поддержка, документация и локализация, а также бэк-офис.

Таким образом, компания предоставляет широкий спектр возможностей для специалистов разных направлений — от программистов и аналитиков до маркетологов и технических специалистов.

В целом, посещение ISPsystem оставило очень положительное впечатление. Мы смогли не только узнать о продуктовой стратегии компании и особенностях её работы, но и почувствовать дружескую атмосферу и высокий уровень вовлеченности сотрудников. Это яркий пример IT-компании, где ценится профессионализм, командная работа и стремление развиваться вместе с продуктом.

## Заключение

В ходе прохождения практики были успешно закреплены знания, полученные в рамках дисциплин «Информатика» и «Программирование». Особое внимание уделялось совершенствованию навыков программирования на языке C++, а также работе с аппаратной платформой Arduino и различными модулями, что позволило глубже понять принципы взаимодействия программного и аппаратного обеспечения.

Также была выполнена задача, связанная с машинным зрением: разрабатывались программы на языке Python с использованием библиотек `opencv (cv2)` и `numpy (np)`, что позволило освоить базовые методы обработки изображений и распознавания цветowych объектов.

В рамках практики мы также посетили две организации: **ISPsystem** и филиал АО «СО ЕЭС» Иркутское РДУ. Эти экскурсии позволили познакомиться с деятельностью современных IT- и энергетических компаний, их технологическими решениями и подходами к организации рабочего процесса.

Полученные знания и практический опыт не только способствовали углублению понимания пройденного материала, но и будут полезны в дальнейшей учебной и профессиональной деятельности.

## Список литературы

1. Wokwi Arduino Simulator: онлайн-симулятор Arduino-проектов [Электронный ресурс]. – URL: <https://wokwi.com/arduino>
2. AlexGyver: подробное руководство по работе со светодиодными лентами WS2812B [Электронный ресурс]. – URL: [https://alexgyver.ru/ws2812\\_guide](https://alexgyver.ru/ws2812_guide)
3. AlexGyver: руководство по работе со светодиодными матрицами [Электронный ресурс]. – URL: [https://alexgyver.ru/matrix\\_guide](https://alexgyver.ru/matrix_guide)
4. AlexGyver: работа с аналоговыми входами Arduino [Электронный ресурс]. – URL: <https://alexgyver.ru/lessons/analog-pins>
5. AlexGyver: работа с последовательным портом Serial [Электронный ресурс]. – URL: <https://alexgyver.ru/lessons/serial>
6. AlexGyver: подключение и управление сервоприводами [Электронный ресурс]. – URL: <https://alexgyver.ru/lessons/servo>
7. Стуков И. Компьютерное зрение OpenCV: где применяется и как работает в Python [Электронный ресурс] / И. Стуков. — Skillbox, 2023. — URL: <https://skillbox.ru/media/code/kompyuternoe-zrenie-opencv-gde-primenyaetsya-i-kak-rabotaet-v-python/>