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 Губарь А.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

(ФИО, должность, подпись)

Оценка по практике

(ФИО, подпись, дата)

Содержание отчета на \_\_\_ стр.

Приложение к отчету на \_\_\_ стр.

Иркутск 2025

**Индивидуальное задание на прохождение**

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| для | Губарь Алисы Александровны | | | | |
|  | (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(а), задание принято к исполнению

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

(подпись

**ДНЕВНИК**

прохождения практики

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| обучающегося | | | | | Губарь Алисы Александровны,  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 | 16-19.06.2025 | Изучала теоретический материал для решения задач №1-3. |  |
| 2 | 20.06.2025 | Решила задачу №1,  Решила задачу №2,  Решила задачу №3,  Начала изучение материала для решения задачи №4. |  |
| 3 | 21.06.2025 | Решила задачу № 4, принялась к изучению материала и к попыткам решения задач №6 и №7. |  |
| 4 | 22.06.2025 | Решила задачу №6,  Решила задачу №7,  Начала собирать схему для задачи №8. |  |
| 5 | 23.06.2025 | Решила задачу №5,  Решила задачу №8,  Начала изучение материала для решения задачи №11. |  |
| 6 | 24.06.2025 | Решила задачу №9  Решила задачу №10. |  |
| 7 | 25.06.2025 | Изучение теоретического материала для решения задачи №11. |  |
| 8 | 26.06.2025 | Решила задачу №11. |  |

|  |  |
| --- | --- |
| Дата фактического прибытия |  |
| обучающегося в структурное подразделение | 16.06.2025 |
| Дата фактического убытия |  |
| обучающегося из структурного подразделения | 29.06.2025 |

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

Содержание

**Введение** ………………………………………………………………………….6

**Задание №1** ………………………………………………………………..……..7

**Задание №2** ………………………………………………………………………9

**Задание №3** ……………………………………………………………………...11

**Задание №4** ……………………………………………………………………...14

**Задание №5** …………………………………………………………………...…16

**Задание №6** ……………………………………………………………………...18

**Задание №7** ……………………………………………………………………...19

**Задание №8** ……………………………………………………………………...20

**Задание №9** ……………………………………………………………………...23

**Задание №10** …………………………………………………………………….27

**Задание №11** …………………………………………………………………….31

**Заключение** ……………………………………………………………………..34

**Список литературы** ……………………………………………………………35

**Введение**

Учебная практика была организована на базе Федерального государственного бюджетного образовательного учреждения высшего образования «Иркутский национальный исследовательский технический университет» и проводилась в период с 16 по 28 июня 2025 года. Главной целью практики стало закрепление теоретических знаний, приобретенных при изучении профильных дисциплин, а также формирование умений их практического применения.

Основной акцент был сделан на освоение программирования на языке C++, развитию способностей к анализу и решению прикладных задач, а также самостоятельной работе с новыми материалами. В рамках практики было выполнено одиннадцать заданий, каждое из которых было ориентировано на отработку определенных тем: от алгоритмов и структур данных до изучения инструментария, применяемого в сфере информационных технологий.

Практика позволила углубить профессиональные компетенции и предоставила возможность реализовать теоретические знания в практической работе.

**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. *Ввод данных*:читаем высоту пирамиды и заполняем пирамиду числами.
2. *Динамическое программирование*: Создаём копию пирамиды и для каждого уровня обновляем *sum.*
3. *Восстановление пути*: начиная с вершины и на каждом уровне идем в направлении меньшей суммы.
4. *Вывод*: последовательность чисел оптимального пути.

**Код программы:**

#include <iostream>

#include <vector>

int main() {

int n;

std::cout << "Введите число: ";

std::cin>>n;

std::vector<std::vector<int>> pyr(n);

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

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

pyr[i].push\_back(std::rand() % 1000);

}

}

std::cout << "Пирамида: " << '\n';

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

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

std::cout << pyr[i][j] << " ";

}

std::cout << '\n';

}

std::vector<std::vector<int>> sum = pyr;

for (int i = n - 2; i >= 0; i--) {

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

sum[i][j] += std::min(sum[i+1][j], sum[i+1][j+1]);

}

}

std::cout << "Минимальная сумма: " << sum[0][0] << '\n';

std::cout << pyr[0][0] << " ";

int j = 0;

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

if (sum[i][j] < sum[i][j+1]) {

std::cout << pyr[i][j] << " ";

} else {

std::cout << pyr[i][j+1] << " ";

j++;

}

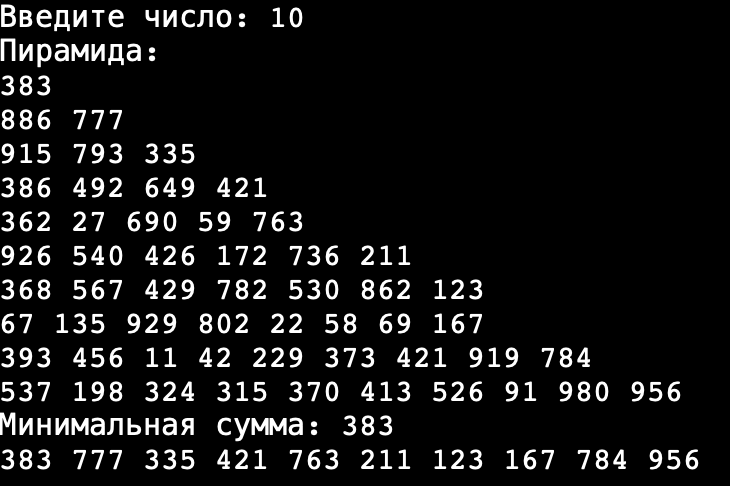
}

return 0;

}

**Таблица тестов:**

|  |  |  |
| --- | --- | --- |
| Ввод | 6 | 3 |
| Вывод | Пирамида:  383  886 777  915 793 335  386 492 649 421  362 27 690 59 763  926 540 426 172 736 211  Минимальная сумма: 383  383 777 335 421 59 172 | Пирамида:  383  886 777  915 793 335  Минимальная сумма: 383  383 777 335 |

****

*Рисунок 1 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. ***Ввод данных*:** читаются числа n (узлы) и m (существующие линии связи) и инициализируется структура DSU для n узлов.
2. *Обработка ребер*: для каждой линии связи (u, v) вызывается dsu.unite(u, v), что объединяет узлы в одно множество.
3. *Подсчёт компонент связности:* число компонент равно количеству корней (parent[i] == i).
4. *Вывод результата:* минимальное число дополнительных рёбер = (число компонент) - 1.

**Код программы:**

#include <iostream>

#include <vector>

#include <numeric>

struct DSU {

std::vector<int> parent;

DSU(int n) {

parent.resize(n + 1);

std::iota(parent.begin(), parent.end(), 0);

}

int find(int i) {

if (parent[i] == i)

return i;

return parent[i] = find(parent[i]);

}

void unite(int i, int j) {

int root\_i = find(i);

int root\_j = find(j);

if (root\_i != root\_j)

parent[root\_i] = root\_j;

}

};

int main() {

std::ios\_base::sync\_with\_stdio(false);

std::cin.tie(NULL);

int n, m;

std::cin >> n >> m;

DSU dsu(n);

for (int k = 0; k < m; ++k) {

int u, v;

std::cin >> u >> v;

dsu.unite(u, v);

}

int components = 0;

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

if (dsu.parent[i] == i)

components++;

}

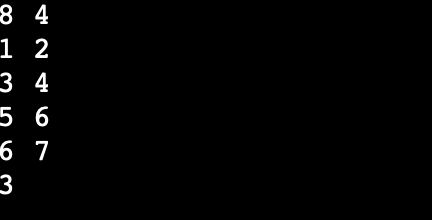
std::cout << components - 1 << std::endl;

return 0;

}

**Таблица тестов:**

|  |  |  |
| --- | --- | --- |
| Ввод | 3 2  10 4  5 7 | 7 4  1 2  2 3  4 5  6 7 |
| Вывод | 2 | 2 |



*Рисунок 2 пример теста из задачи №2*

**Задание №3**

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.

**Алгоритм программы:**

1. ***Ввод данных*: ч**итаются n (количество улиц) и k (количество операций).
2. *Цикл обработки операций*:   
   Тип 1 (1 i x):

Вызов ft.update(i, x) для добавления x мм снега на улицу i.

Тип 2 (2 u r):

Вычисление суммы снега на улицах от u до r как ft.query(r) - ft.query(u - 1).

Результат сохраняется в вектор results.

1. *Вывод результатов:* после обработки всех операций выводятся сохраненные результаты запросов.

**Код программы:**

#include <iostream>

#include <vector>

class FenwickTree {

private:

std::vector<long long> bit;

int size;

public:

FenwickTree(int n) : size(n), bit(n + 1, 0) {}

void update(int idx, int delta) {

for (; idx <= size; idx += idx & -idx) {

bit[idx] += delta;

}

}

long long query(int idx) {

long long sum = 0;

for (; idx > 0; idx -= idx & -idx) {

sum += bit[idx];

}

return sum;

}

};

int main() {

std::ios\_base::sync\_with\_stdio(false);

std::cin.tie(NULL);

int n, k;

std::cin >> n >> k;

FenwickTree ft(n);

std::vector<long long> results;

for (int op = 0; op < k; ++op) {

int type;

std::cin >> type;

if (type == 1) {

int i, x;

std::cin >> i >> x;

ft.update(i, x);

} else {

int u, r;

std::cin >> u >> r;

results.push\_back(ft.query(r) - ft.query(u - 1));

}

}

for (long long res : results) {

std::cout << res << std::endl;

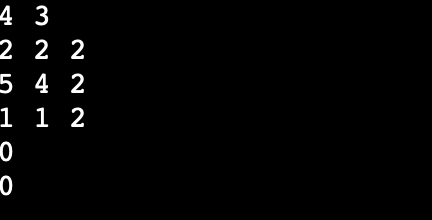
}

return 0;

}

**Таблица тестов:**

|  |  |  |
| --- | --- | --- |
| Ввод | 6 5  2 1 6  1 3 2  2 2 4  1 6 3  2 1 6 | 5 3  1 3 7  1 1 4  2 2 5 |
| Вывод | 0  2  5 | 7 |



*Рисунок 3 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 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. ***Построение обратной перестановки***:для заданной перестановки P длиной n создается обратная перестановка это позволяет "отменить" действие исходной перестановки.
2. *Применение перестановки к строке*: функция apply\_permutation перемещает символы строки согласно перестановке и для каждого индекса i в результирующей строке берется символ из позиции P[i] исходной строки.
3. *Многократное обратное преобразование*: для дешифровки k раз применяется обратная перестановка к зашифрованной строке, каждое применение соответствует одному шагу дешифровки.

**Код программы:**

#include <iostream>

#include <vector>

#include <string>

using namespace std;

vector<int> inverse\_permutation(const vector<int>& P) {

int n = P.size();

vector<int> P\_inv(n);

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

P\_inv[P[i] - 1] = i;

}

return P\_inv;

}

string apply\_permutation(const string& s, const vector<int>& P) {

int n = s.size();

string result(n, ' ');

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

result[i] = s[P[i]];

}

return result;

}

string decrypt(const string& encrypted, const vector<int>& P, int k) {

int n = encrypted.size();

vector<int> P\_inv = inverse\_permutation(P);

string current = encrypted;

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

current = apply\_permutation(current, P\_inv);

}

return current;

}

int main() {

int n, k;

cin >> n >> k;

vector<int> P(n);

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

cin >> P[i];

}

string encrypted;

cin.ignore();

getline(cin, encrypted);

string original = decrypt(encrypted, P, k);

cout << original << endl;

return 0;

}

**Таблица тестов:**

|  |  |  |
| --- | --- | --- |
| Ввод | 6 5  2 1 6  1 3 2  2 2 4  1 6 3  2 1 6 |  |
| Вывод | 0  2  5 |  |

**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:* чтение размеров матрицы, заполнение матрицы значениями 0 и 1.**
2. *Поиск границ*: инициализация переменных, обход матрицы: при нахождении 1 обновляются границы.

**Код программы:**

#include <iostream>

#include <vector>

#include <climits>

using namespace std;

int main() {

int h, w;

cin >> h >> w;

vector<vector<int>> matrix(h, vector<int>(w));

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

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

cin >> matrix[i][j];

}

}

int min\_row = INT\_MAX, min\_col = INT\_MAX;

int max\_row = INT\_MIN, max\_col = INT\_MIN;

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

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

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;

}

}

}

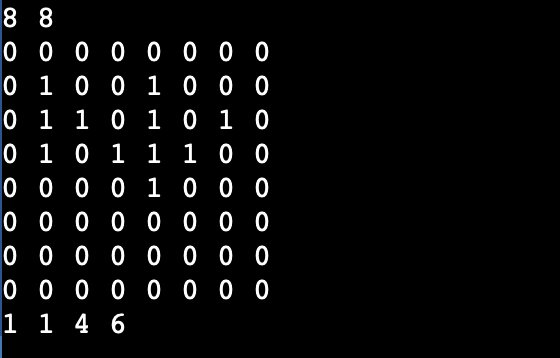
cout << min\_row << " " << min\_col << " " << max\_row << " " << max\_col << endl;

return 0;

}

**Таблица тестов:**

|  |  |  |
| --- | --- | --- |
| Ввод | 5 5  0 0 0 0 0  0 1 1 1 0  0 1 1 1 0  0 1 1 1 0  0 0 0 0 0 | 6 6  0 0 0 0 0 0  0 0 1 1 0 0  0 0 1 1 0 0  0 0 1 1 0 0  0 0 0 0 0 0  0 0 0 0 0 0 |
| Вывод | 1 1 3 3 | 1 2 3 3 |

****

*Рисунок 4 пример теста из задачи №5*

**Задание №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. ***Ввод данных:* считывается количество запросов и для каждого запроса считываются 4 числа: количество контроллеров и модулей.**
2. *Вычисление максимального числа пар.*
3. *Результаты для всех запросов сохраняются в список:* Список преобразуется в строку с разделителем-пробелом.

**Код программы:**

n=int(input())

otvet=[]

for i in range (n):

a,b,x,y=map(int, input().split())

otvet.append(min(b,x)+abs(max(b,x)-(min(b,x)))+min((a - abs(max(b,x)-(min(b,x)))), y))

otvet=[str(x) for x in otvet]

print(' '.join(otvet))

**Таблица тестов:**

|  |  |  |
| --- | --- | --- |
| Ввод | 2  3 2 4 1  5 1 3 2 | 3  5 3 4 2  2 4 1 3  7 0 5 3 |
| Вывод | 3 3 | 5 3 3 |



*Рисунок.5 пример теста из задачи №6*

**Задание №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. ***Ввод данных*: п**олучить входную строку s из 6 символов
2. *Цикл обработки операций*: 1. Определить множество допустимых заглавных латинских букв a, если первый символ s[0] не содержится в множестве a – вывести «no» и завершить выполнение. 2. П**роверка цифровой части (символы 1-3): о**пределить множество цифр b, если любой из символов s[1], s[2] или s[3] не содержится в множестве b, вывести «no» и завершить выполнение. 3. П**роверка буквенной части (символы 4-5): Е**сли любой из символов s[4] или s[5] не содержится в множестве a, вывести «no» и завершить выполнение.
3. *Вывод результатов:* если все проверки пройдены успешно вывести «yes».

**Код программы:**

s=input()

a="QWERTYUIOPASDFGHJKLZXCVBNM"

b="0123456789"

if s[0] not in a: print ('no')

else:

if s[1] not in b or s[2] not in b or s[3] not in b:print ('no')

else:

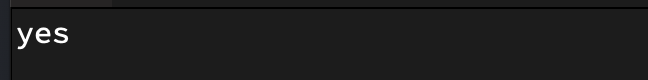
if s[4] not in a or s[5] not in a:

print ('no')

else: print('yes')

**Таблица тестов:**

|  |  |  |
| --- | --- | --- |
| Ввод | P142EQ | 1W45OP |
| Вывод | yes | no |



*Рисунок 6 пример теста из задачи №7*

**Задание №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 и конфигурация*: о**пределение параметров матрицы: ширина (8), высота (8), общее количество светодиодов (64); указание пина подключения (D3); установка яркости на средний уровень, инициализация последовательного порта.
2. *Преобразование координат*: четные строки - прямой порядок нумерации, нечетные строки - обратный порядок (змеевидная разводка).
3. *Основной цикл*: 1.Бегущая точка *-* Поочередное включение каждого светодиода синим цветом, остальные светодиоды при этом отключены, скорость движения: 50 мс на точку. 2. Цветовые волны - плавное изменение цветов по всей матрице, постепенное изменение палитры цветов. 3. Случайные вспышки – одновременное появление 10 случайных светодиодов случайного цвета. Короткая пауза между обновлениями.

**Код программы:**

#include <FastLED.h>

#define LED\_PIN

#define MATRIX\_WIDTH 8

#define MATRIX\_HEIGHT 8

#define NUM\_LEDS (MATRIX\_WIDTH \* MATRIX\_HEIGHT)

CRGB leds[NUM\_LEDS];

int getLedIndex(int x, int y) {

if (y % 2 == 0) {

return y \* MATRIX\_WIDTH + x;

} else {

return y \* MATRIX\_WIDTH + (MATRIX\_WIDTH - 1 - x);

}

}

void setup() {

FastLED.addLeds<WS2812B, LED\_PIN, GRB>(leds, NUM\_LEDS);

FastLED.setBrightness(50);

**Serial**.begin(9600);

}

void loop() {

runningDot();

colorWaves();

randomSparkles();

}

void runningDot() {

for(int i = 0; i < NUM\_LEDS; i++) {

// Гасим все светодиоды

fill\_solid(leds, NUM\_LEDS, CRGB::Black);

leds[i] = CRGB::Blue;

FastLED.show();

delay(50);

}

}

void colorWaves() {

static uint8\_t hue = 0;

for(int x = 0; x < MATRIX\_WIDTH; x++) {

for(int y = 0; y < MATRIX\_HEIGHT; y++) {

int index = getLedIndex(x, y);

leds[index] = CHSV(hue + (x \* 10) + (y \* 10), 255, 255);

}

}

hue += 5;

FastLED.show();

delay(50);

}

void randomSparkles() {

fill\_solid(leds, NUM\_LEDS, CRGB::Black);

for(int i = 0; i < 10; i++) {

int pos = random(NUM\_LEDS);

leds[pos] = CHSV(random(255), 255, 255);

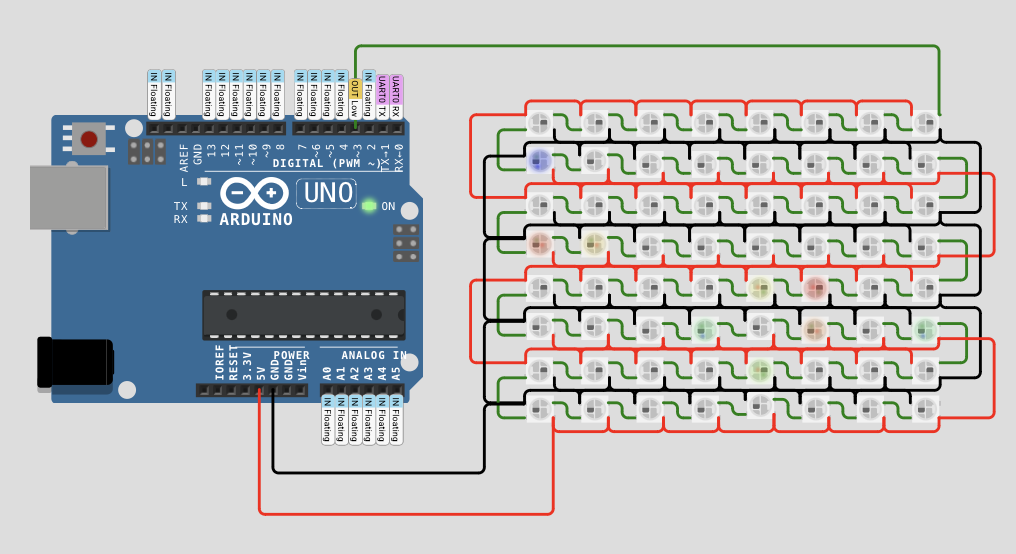
}

FastLED.show();

delay(200);

}

**Пример работы программы:**



*Рисунок 7 Вспышка случайных цветов*

**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.

Схема приведена на рисунке 1.

Зеленый светодиод – двери отрываются.

Красный светодиод ­– двери закрываются.

Фоторезистор имитируют процесс приближения-удаления человека от дверей.

Изменение значений фоторезистора осуществляется при помощи ползунка (рисунок 2), изменение значения фоторезистора доступно только, когда запущен процесс моделирования.

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.

**Алгоритм программы:**

1. ***Инициализация системы*: на**стройка пинов: красный светодиод (9), зеленый светодиод (5), фоторезистор (A0); установка времени открытия дверей: 5 секунд; начальное состояние: двери закрыты (красный свет горит); запуск последовательного порта для мониторинга.
2. *Основной цикл*: 1. Считывание данных - чтение значения с фоторезистора (уровень освещенности). 2. Проверка условия открытия - если освещенность высокая (значение > 512) И двери закрыты:  
    - выполнить открытие дверей

- зафиксировать время открытия

- включить зеленый светодиод

- выключить красный светодиод

3. Проверка условия закрытия - если двери открыты И прошло 5 секунд, то повторно проверить освещенность и если освещеннность низкая (значение ≤ 512):

- закрыть двери

- включить красный светодиод

- выключить зеленый светодиод

Если освещенность остается высокой, то продлить время открытия и сообщить и продолжении работы. 4. Задержка 100 мс для стабильности работы.

**Код программы:**

// Определение пинов для светодиодов и фоторезистора

const int redLedPin = 9; // Красный светодиод (двери закрыты)

const int greenLedPin = 5; // Зеленый светодиод (двери открыты)

const int photoResistorPin = A0; // Аналоговый пин для фоторезистора

// Временной интервал, на который открываются двери (в миллисекундах)

const unsigned long doorOpenTime = 5000; // 5 секунд

// Переменные для управления состоянием

unsigned long doorOpenStartTime = 0;

bool doorIsOpen = false;

void setup() {

// Инициализация пинов светодиодов как выходов

pinMode(redLedPin, OUTPUT);

pinMode(greenLedPin, OUTPUT);

// По умолчанию двери закрыты (горит красный светодиод)

digitalWrite(redLedPin, HIGH);

digitalWrite(greenLedPin, LOW);

// Инициализация последовательного порта для отладки

**Serial**.begin(9600);

**Serial**.println("Система автоматических дверей запущена");

**Serial**.println("Двери закрыты");

}

void loop() {

// Чтение значения с фоторезистора

int sensorValue = analogRead(photoResistorPin);

// Если значение превышает порог и двери еще не открыты

if (sensorValue > 512 && !doorIsOpen) {

openDoor();

}

// Если двери открыты и прошло достаточно времени

if (doorIsOpen && (millis() - doorOpenStartTime >= doorOpenTime)) {

// Проверяем значение фоторезистора снова

sensorValue = analogRead(photoResistorPin);

if (sensorValue <= 512) {

closeDoor();

} else {

// Обновляем время открытия, чтобы проверить снова через doorOpenTime

doorOpenStartTime = millis();

**Serial**.println("Двери остаются открытыми, человек все еще рядом");

}

}

// Небольшая задержка для стабильности

delay(100);

}

// Функция для открытия дверей

void openDoor() {

digitalWrite(redLedPin, LOW);

digitalWrite(greenLedPin, HIGH);

doorIsOpen = true;

doorOpenStartTime = millis();

**Serial**.println("Двери открываются!");

}

// Функция для закрытия дверей

void closeDoor() {

digitalWrite(greenLedPin, LOW);

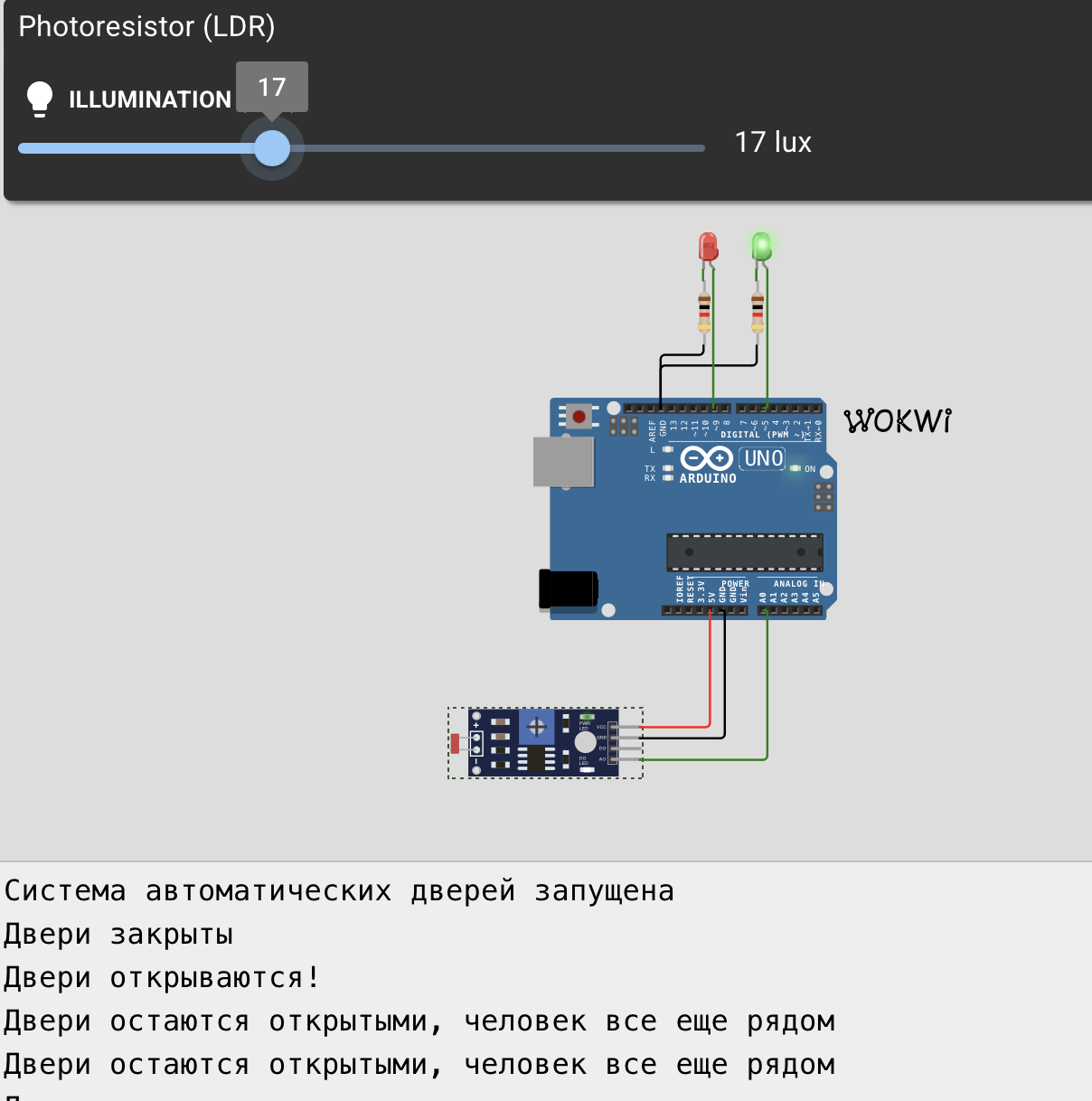
digitalWrite(redLedPin, HIGH);

doorIsOpen = false;

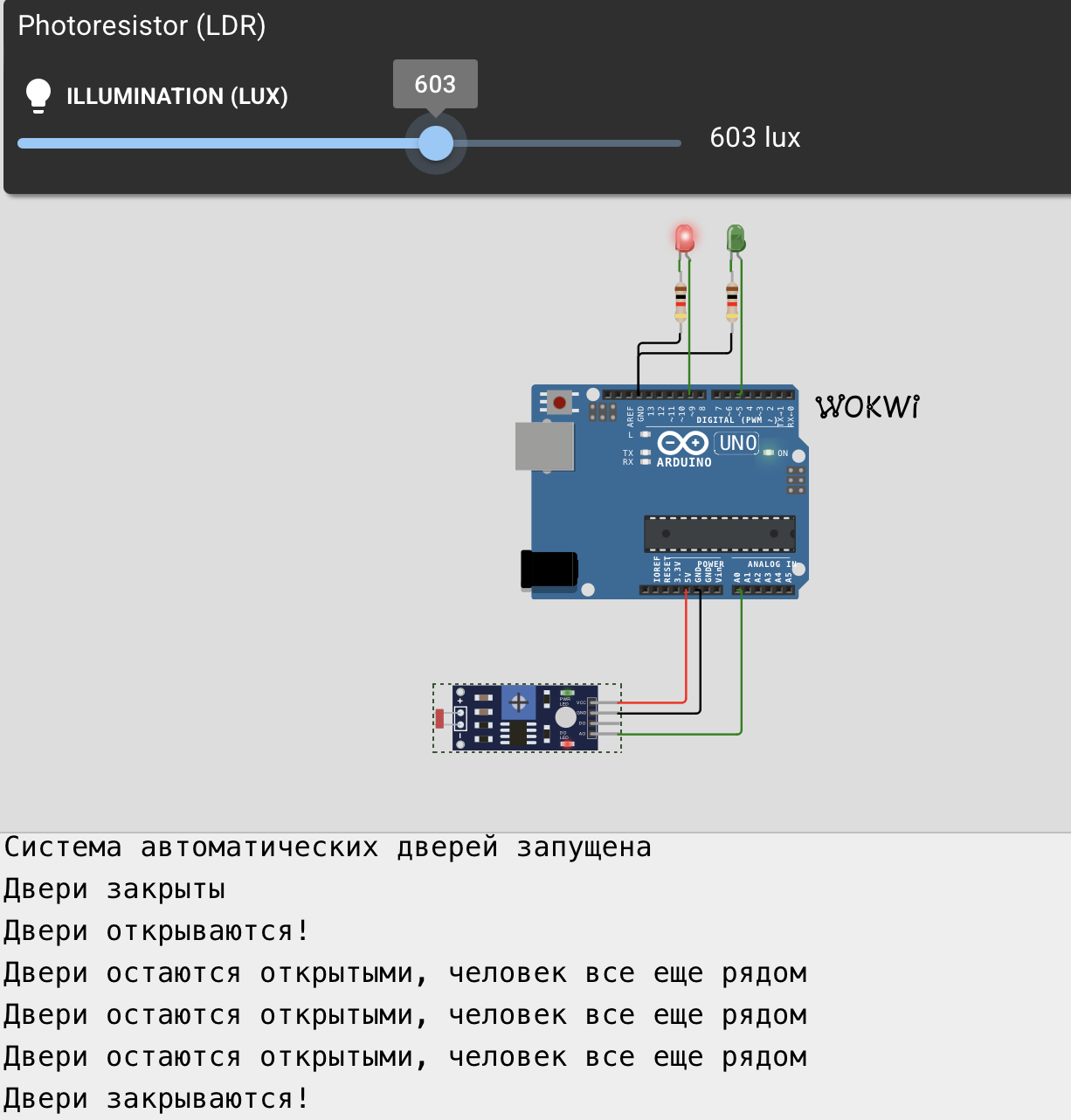
**Serial**.println("Двери закрываются!");

}

**Пример работы программы:**



*Рисунок 8 Зеленый светодиод*



*Рисунок 9 Красный светодиод*

**Задание №10**

**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.

**Алгоритм программы:**

1. ***Инициализация системы*: п**одключение сервопривода к пину 3, установка начального положения (0 градусов), запуск последовательного порта для команд
2. *Основной цикл*: 1. Проверка входящих команд - постоянная проверка наличия данных в Serial Monitor, если получена команда:

- считывание строки ввода

- проверка корректности ввода (только цифры)

- проверка диапазона (0-180 градусов)

2. Обработка команды – если ввод корректен и угол верный, то происходит установка целевого угла и подтверждение команды. Если угол тот же – сообщение о текущем положении. Если ошибка ввода – сообщение об ошибке. 3. Плавное перемещение:

- постепенное изменение текущего угла к целевому

- шаг изменения: 1 градус

- задержка 15 мс для плавности движения

- обновление положения сервопривода

**Код программы:**

#include <Servo.h>

Servo myServo; // Создаем объект сервопривода

const int servoPin = 3; // Пин для подключения сервопривода

int currentAngle = 0; // Текущий угол сервопривода (начальное положение)

int targetAngle = 0; // Целевой угол сервопривода

void setup() {

myServo.attach(servoPin); // Подключаем сервопривод к указанному пину

myServo.write(currentAngle); // Устанавливаем начальное положение

**Serial**.begin(9600); // Инициализация последовательного порта

**Serial**.println("Система управления сервоприводом готова");

**Serial**.println("Введите угол от 0 до 180 градусов:");

}

void loop() {

// Проверяем наличие данных в последовательном порту

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

String input = **Serial**.readStringUntil('\n'); // Читаем строку до символа новой строки

input.trim(); // Удаляем лишние пробелы

// Проверяем, является ли ввод числом

if (isValidNumber(input)) {

int newAngle = input.toInt(); // Преобразуем строку в число

// Проверяем, находится ли угол в допустимом диапазоне

if (newAngle >= 0 && newAngle <= 180) {

if (newAngle != currentAngle) {

targetAngle = newAngle;

**Serial**.print("Поворачиваю сервопривод на угол: ");

**Serial**.println(targetAngle);

} else {

**Serial**.println("Сервопривод уже в этом положении");

}

} else {

**Serial**.println("Ошибка: угол должен быть от 0 до 180 градусов");

}

} else {

**Serial**.println("Ошибка: введите числовое значение от 0 до 180");

}

}

// Плавное перемещение сервопривода к целевому углу

if (currentAngle != targetAngle) {

if (currentAngle < targetAngle) {

currentAngle++;

} else {

currentAngle--;

}

myServo.write(currentAngle);

delay(15); // Задержка для плавности движения

}

}

// Функция проверки, является ли строка числом

bool isValidNumber(String str) {

for (byte i = 0; i < str.length(); i++) {

if (!isDigit(str.charAt(i))) {

return false;

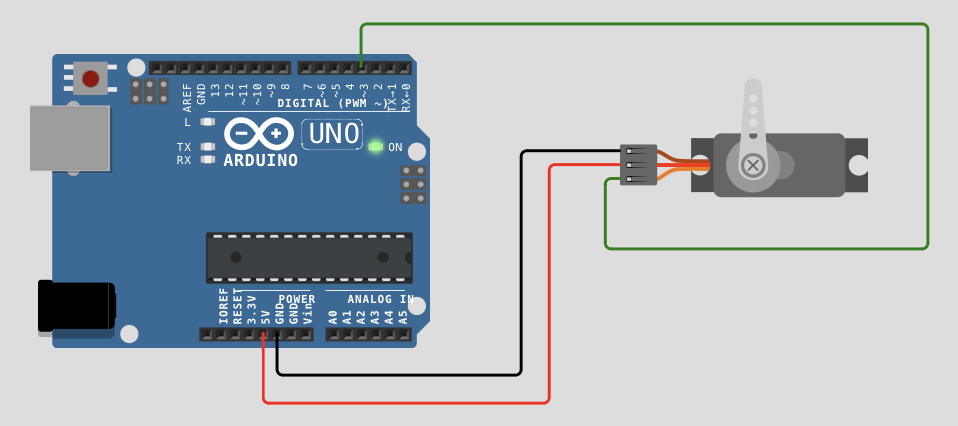
}

}

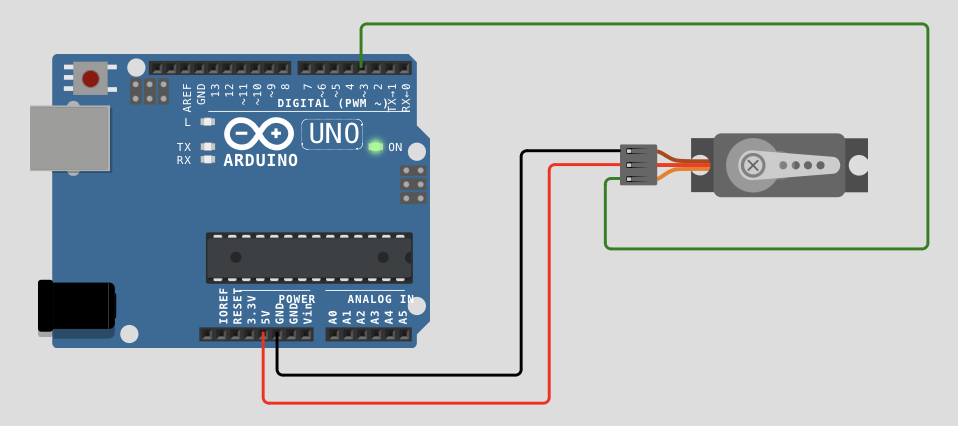
return str.length() > 0;

}

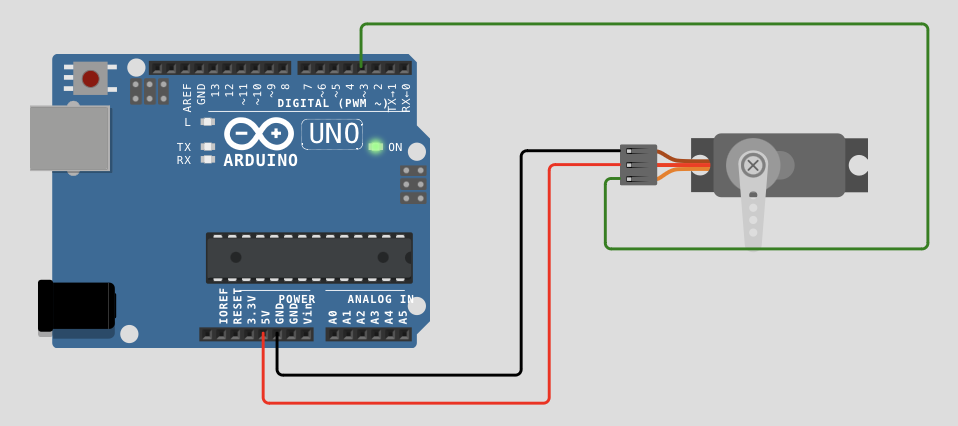
**Пример работы программы:**



*Рисунок 10 Положение на 0 градусов*

****

*Рисунок 11 Положение на 90 градусов*

****

*Рисунок 12 Положение на 180 градусов*

**Задание №11**

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

Перевод:

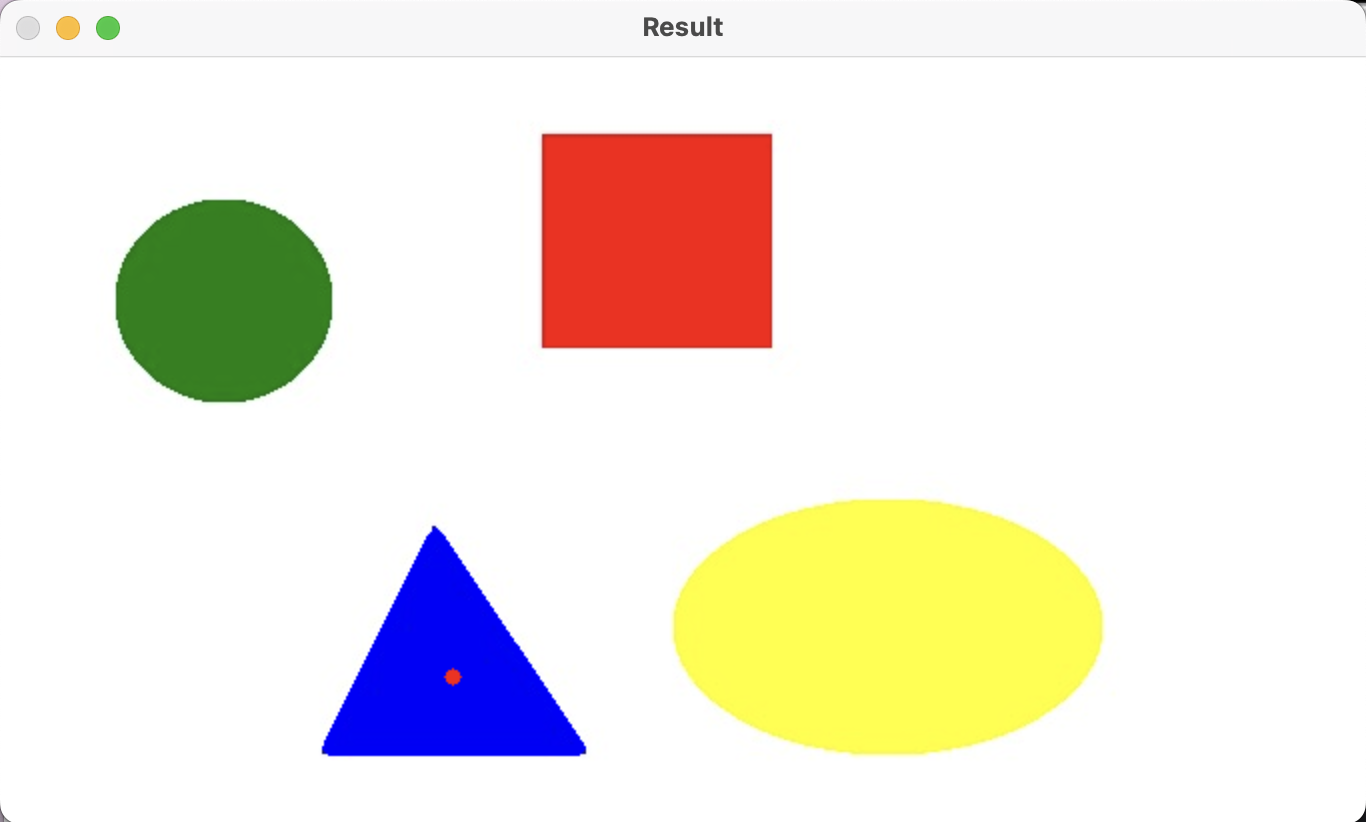
Найдите все синие объекты в изображении. Найдите центры синих объектов. Отметьте центр красной точкой.

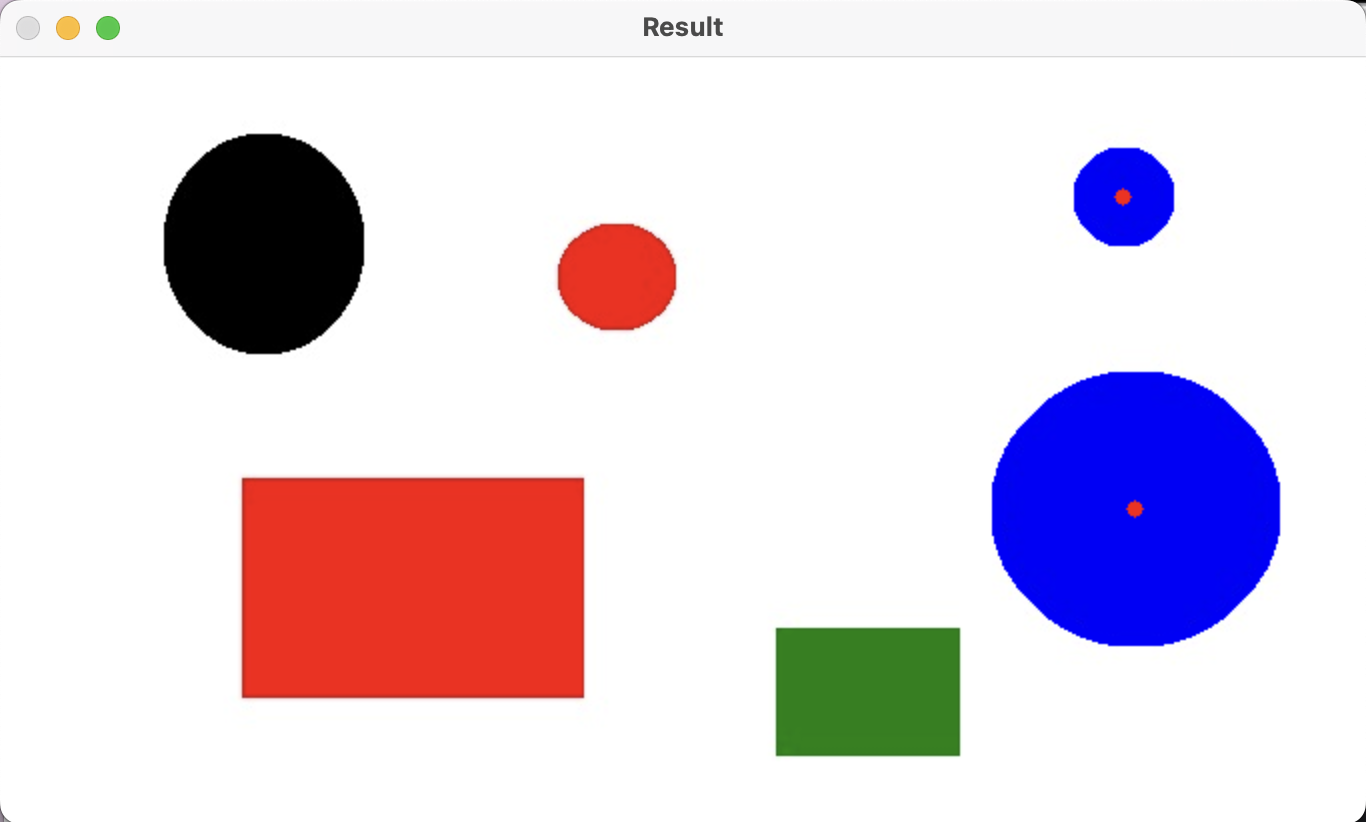
**Алгоритм программы:**

1. ***Загрузка изображения***
2. *Предобработка:* к изображению применяется размытие для уменьшения шума. Преобразование изображения из пространства BGR в HSV для более удобной работы с цветами.
3. *Выделение синего цвета:* задается нижняя и верхняя граница синего цвета в HSV. Создается бинарная маска (cv2.inRange), где белые пиксели соответствуют найденным синим областям.
4. *Поиск контуров:* находятся все контуры и их иерархия (cv2.findContours). Иерархия позволяет определить вложенность контуров (например, дырки внутри фигур).
5. *Фильтрация контуров:* перебор контуров. Остаются только внешние контуры (те, у которых родитель равен -1 в иерархии).
6. *Нахождение центра фигуры:* для каждого внешнего контура строится минимальная описывающая окружность (cv2.minEnclosingCircle). Центр окружности принимается за центр объекта.
7. *Отрисовка:* в центре объекта рисуется красная точка (cv2.circle). Сам контур выделяется синим (cv2.drawContours).
8. *Вывод результатов:* показывается обработанное изображение в отдельном окне.

**Код программы:**

import cv2  
import numpy as np  
  
# Загрузка изображения  
image = cv2.imread("/Users/pawwience/Downloads/Безымянный.jpg")  
  
if image is None:  
 print("Ошибка: изображение не загружено. Проверь путь к файлу.")  
 exit()  
  
# Размытие  
blurred = cv2.GaussianBlur(image, (11, 11), 0)  
  
# Перевод в HSV  
hsv = cv2.cvtColor(blurred, cv2.COLOR\_BGR2HSV)  
  
# Диапазон синего  
blue\_min = np.array([100, 150, 50], np.uint8)  
blue\_max = np.array([140, 255, 255], np.uint8)  
blue\_mask = cv2.inRange(hsv, blue\_min, blue\_max)  
  
# Контуры  
blue\_contours, blue\_hierarchy = cv2.findContours(  
 blue\_mask.copy(), cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE  
)  
  
if blue\_hierarchy is not None:  
 for i, contour in enumerate(blue\_contours):  
 if blue\_hierarchy[0][i][3] == -1: # внешний контур  
 # Минимальная окружность  
 (x, y), radius = cv2.minEnclosingCircle(contour)  
 center = (int(x), int(y))  
  
 # Отметить центр красной точкой  
 cv2.circle(image, center, 4, (0, 0, 255), -1)  
  
 # Обвести контур  
 cv2.drawContours(image, [contour], -1, (255, 0, 0), 3)  
  
# Показ результата  
cv2.imshow("Result", image)  
cv2.waitKey(0)  
cv2.destroyAllWindows()

  
*Рисунок 13 Пример работы программы*



*Рисунок 14 Пример работы программы*

# **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 цветовых объектов.

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
3. AlexGyver: руководство по работе со светодиодными матрицами [Электронный ресурс]. – URL: 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>