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 Симонов А. А., ИСИб-24-1

(ФИО, группа, подпись)

Руководитель практики от института ИТиАД

Кононенко Роман Владимирович, доцент

(ФИО, должность, подпись)

Ссылка на резюме обучающего на сайте https://www.superjob.ru/

Руководитель образовательной программы

Кононенко Р.В., доцент института ИТиАД

(ФИО, должность, подпись)



Оценка по практике

(ФИО, подпись, дата)

Содержание отчета на 4 стр.

Иркутск 2025

Ссылка на резюме обучающего на сайте https://www.hh.ru/

**Индивидуальное задание на прохождение**

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| для | Симонова Александра Андреевича | | | | |
|  | (ФИО обучающегося полностью) | | | | |
| обучающегося | | 1 | курса | группы | ИСИб-24-1 |

по направлению подготовки Информационные системы и технологии

профиль Интеллектуальные системы обработки информации и управления

Место прохождения практики: ИРНИТУ

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

Цели и задачи прохождения практики:

Содержание практики, вопросы, подлежащие изучению:

Планируемые результаты практики:

Руководитель практики от

института ИТиАД

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

(подпись

**Согласовано:**

Руководитель ООП

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

(подпись

«\_\_\_»\_\_\_\_\_\_\_\_\_\_2023 г.

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

\_\_\_\_\_\_\_\_\_\_«16» июня 2025 г.

(подпись

**ДНЕВНИК**

прохождения практики

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

Иркутск 2025

Руководителем практики от структурного подразделения назначен:

Кононенко Роман Владимирович, доцент

(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 | Выполнена задача №1, задача №2 |  |
| 2 | 17.06.2025 | Выполнена задача №3 |  |
| 3 | 18.06.2025 | Выполнена задача №4, задача №5, задача №6 |  |
| 4 | 19.06.2025 | Выполнена задача №7 |  |
| 4 | 20.06.2025 | Выполнена задача №9, задача №10 |  |
| 4 | 22.06.2025 | Выполнена задача №8 |  |
| 4 | 23.06.2025 | Выполнена задача №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) |

**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. Пример наиболее короткого маршрута выделен красным цветом, сумма чисел = 10.



Рисунок 1 – пример горы

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

**Формат входных данных**

В первой строке дано целое число N - высота пирамиды, далее следуют N строк из чисел, разделённых пробелом (в каждой строке на 1 число больше, чем в предыдущей)

**Формат выходных данных**

Сумма чисел в пути с вершины до основания (одно число)

Последовательность участков маршрута (числа, разделённые пробелом)

Ход решения:

#include <iostream>

#include <vector>

#include <cstdlib>

#include <algorithm>

using namespace std;

vector<vector<int>> hill(int n) {

    vector<vector<int>> pyr(n);

    for (int i = 0; i < n; ++i) {

        pyr[i].resize(i + 1);

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

            pyr[i][j] = rand() % 100 + 1;

        }

    }

    return pyr;

}

pair<int, vector<int>> minPath(const vector<vector<int>>& pyr) {

    int n = pyr.size();

    vector<vector<int>> dp(n, vector<int>(n, 0));

    vector<vector<int>> par(n, vector<int>(n, -1));

    dp[0][0] = pyr[0][0];

    for (int i = 1; i < n; ++i) {

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

            if (j == 0) {

                dp[i][j] = dp[i - 1][j] + pyr[i][j];

                par[i][j] = j;

            } else if (j == i) {

                dp[i][j] = dp[i - 1][j - 1] + pyr[i][j];

                par[i][j] = j - 1;

            } else {

                if (dp[i - 1][j - 1] < dp[i - 1][j]) {

                    dp[i][j] = dp[i - 1][j - 1] + pyr[i][j];

                    par[i][j] = j - 1;

                } else {

                    dp[i][j] = dp[i - 1][j] + pyr[i][j];

                    par[i][j] = j;

                }

            }

        }

    }

    int minSum = dp[n - 1][0], minIdx = 0;

    for (int j = 1; j < n; ++j) {

        if (dp[n - 1][j] < minSum) {

            minSum = dp[n - 1][j];

            minIdx = j;

        }

    }

    vector<int> path;

    for (int i = n - 1; i >= 0; --i) {

        path.push\_back(pyr[i][minIdx]);

        minIdx = par[i][minIdx];

    }

    reverse(path.begin(), path.end());

    return {minSum, path};

}

int main() {

    cout << "Здравствуйте, введите высоту пирамиды: ";

    int n;

    cin >> n;

    rand();

    auto pyr = hill(n);

    for (const auto& row : pyr) {

        for (size\_t i = 0; i < row.size(); ++i) {

            cout << row[i] << (i < row.size() - 1 ? " " : "");

        }

        cout << "\n";

    }

    auto result = minPath(pyr);

    cout << result.first << "\n";

    for (size\_t i = 0; i < result.second.size(); ++i) {

        cout << result.second[i] << (i < result.second.size() - 1 ? " " : "");

    }

    cout << "\n";

    return 0;

}

Результат:

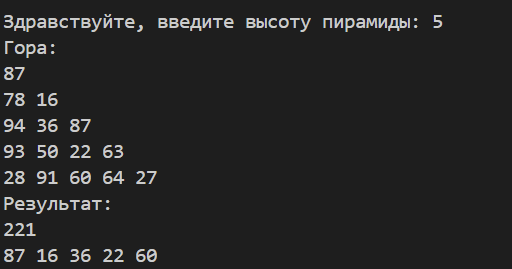


Рисунок 2 – пример работы

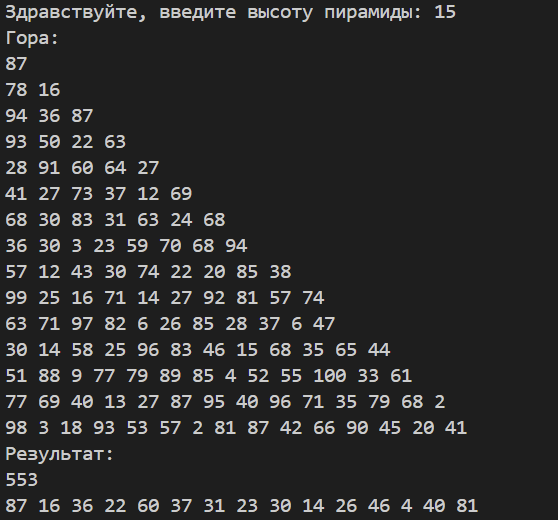


Рисунок 3 – пример работы

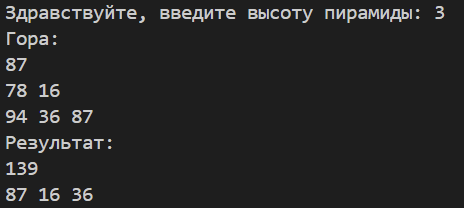


Рисунок 3 – пример работы

**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.



Рисунок 5 – компьютерная сеть с 3 частями

**Формат входных данных**

В первой строке дано целое число N - количество узлов сети и M - число линий связи. Далее следуют M строк из чисел, разделённых пробелом (узлы, которые связывает данная линия)

**Формат выходных данных**

Число необходимых линий связи (одно число)

Ход решения:

#include <iostream>

#include <vector>

using namespace std;

vector<vector<int>> generateGraph(int n, int m) {

    vector<vector<int>> graph(n + 1);

    cout << "Введите " << m << " рёбер (формат: u v, где u и v <= "<< n <<"):" << endl;

    for (int i = 0; i < m; ++i) {

        int u, v;

        cin >> u >> v;

        graph[u].push\_back(v);

        graph[v].push\_back(u);

    }

    return graph;

}

void dfs(int v, vector<bool>& visited, const vector<vector<int>>& graph) {

    visited[v] = true;

    for (int neighbor : graph[v]) {

        if (!visited[neighbor]) {

            dfs(neighbor, visited, graph);

        }

    }

}

int count\_communication\_lines(int n, const vector<vector<int>>& graph) {

    vector<bool> visited(n+1, false);

    int count = -1;

    for (int i = 1; i < n+1; ++i) {

        if (!visited[i]) {

            dfs(i, visited, graph);

            ++count;

        }

    }

    return count;

}

int main() {

  int n, m;

  cout << "Здравствуйте, введите число вершин графа (узлы сети, не более 1000)" << endl <<

  "и набор рёбер (сохранившиеся линии связи, не более 1000) через пробел и нажмите enter:";

  cin >> n;

  cin >> m;

  vector<vector<int>> graph = generateGraph(n,m);

  if (n <= 0 or m <= 0){

    cout << "неверные данные. Пожалуйста, введите количество вершин, превышающее 0!" << endl;

      return 0;

  }

  int count = count\_communication\_lines(n, graph);

  cout << "минимальное число линий связи: "<<count << endl;

  return 0;

}

Результат:

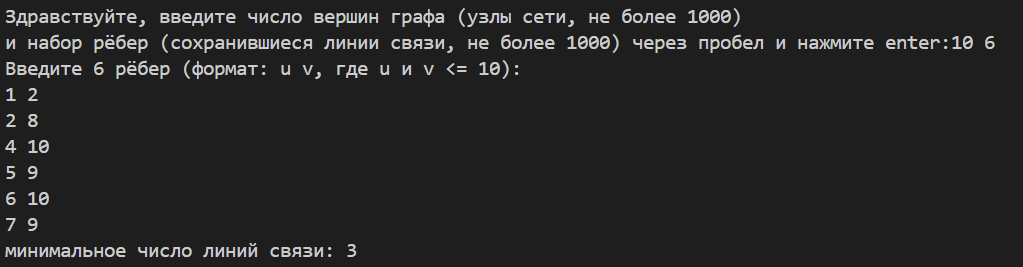


Рисунок 6 – пример работы

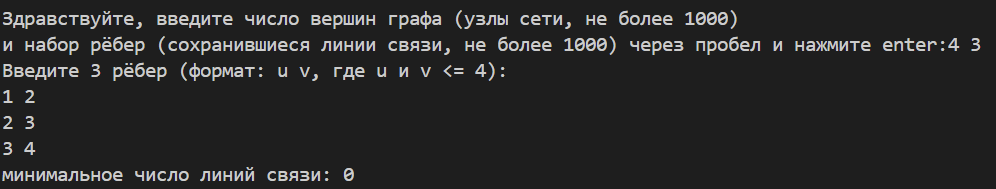


Рисунок 7 – пример работы

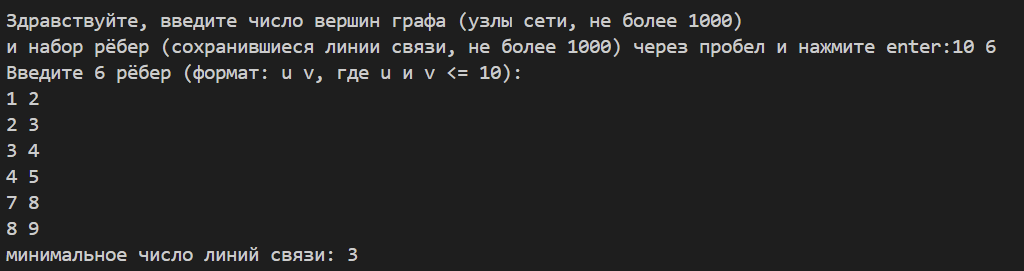


Рисунок 8 – пример работы

**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.

**Формат входных данных**

Первая строка входных данных содержит два целых числа – n (1 или больше) и k (0 или больше) это количество чисел в массиве и количество запросов соответственно.

Следующие k строк содержат:

* либо 1 i x – Учетная информация о количестве, выпавшего на улице i (больше 0) x миллиметров снега.
* либо 2 u r – Запрос на подсчет количества снега на улицах от u до r (u и r больше 0 и могут быть равны друг другу)

**Формат выходных данных**

На каждый запрос второго типа надо вывести единственное число – суммарное выпавшего на них снега с момента начала наблюдения.

Ход решения:

#include <iostream>

#include <vector>

using namespace std;

void snow\_center(int n, int k){

  vector<int> snow\_level(n,0);

  int count\_k = 0;

  int command, a, b;

  while (count\_k != k)

  {

    cin >> command >> a >> b;

    if (command == 1){

      snow\_level[a-1] = b;

    } else if (command == 2){

      int snow\_count = 0;

      for (int i = a-1; i <= b-1; i++){

        snow\_count += snow\_level[i];

      }

      cout << snow\_count << endl;

    } else {

      cout << "Неправильные входные данные. они должны начинаться либо с 1, либо с 2." << endl;

      continue;

    }

    ++count\_k;

  }

}

int main() {

  int n,k;

  cout << "введите количество улиц и число запросов через пробел: ";

  cin >> n >> k;

  snow\_center(n,k);

  return 0;

}

Результат:

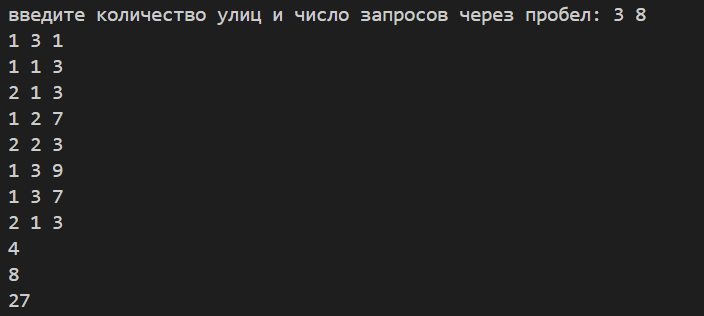


Рисунок 9 – пример работы

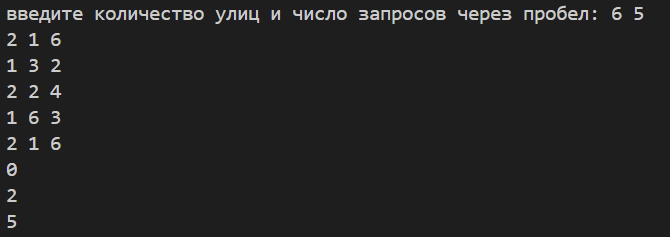


Рисунок 10 – пример работы

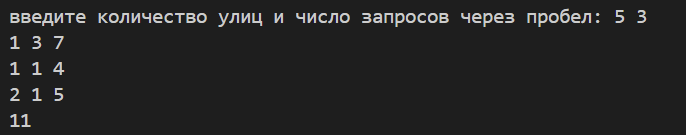


Рисунок 11 – пример работы

**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.



Рисунок 12 – Перестановка слова transposition 2 раза

Вам дано зашифрованное таким образом слово, шифрующая перестановка P и число k. Необходимо восстановить слово.

**Формат входных данных**

Первая строка входных данных содержит 2 числа – n и k (1 или больше, могут быть равны). Следующая строка содержит перестановку длиной n, числа разделяются пробелом. Третья строка содержит зашифрованное слово длиной n.

**Формат выходных данных**

Вывести одну строку − исходное слово.

Ход решения:

#include <iostream>

#include <vector>

#include <string>

using namespace std;

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

    int n = encrypted.size();

    string original = encrypted;

    vector<int> inversePermutation(n);

    for (int i = 0; i < n; ++i) {

        inversePermutation[permutation[i] - 1] = i;

    }

    for (int i = 0; i < k; ++i) {

        string temp = original;

        for (int j = 0; j < n; ++j) {

            original[j] = temp[inversePermutation[j]];

        }

    }

    return original;

}

int main() {

    int n, k;

    cout << "Введите данные:"

    cin >> n >> k;

    vector<int> permutation(n);

    for (int i = 0; i < n; ++i) {

        cin >> permutation[i];

    }

    string encrypted;

    cin >> encrypted;

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

    cout << original << endl;

    return 0;

}

Результат:

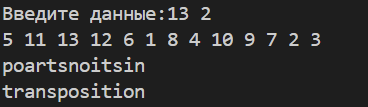


Рисунок 13 – Пример работы

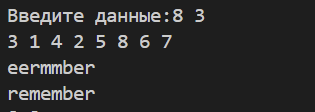


Рисунок 14 – Пример работы

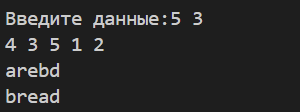


Рисунок 15 – Пример работы

**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.

**Формат входных данных**

В первой строке через пробел заданы высота h и ширина w матрицы (длина и ширина 10 или больше, но не больше 50, могут быть равны). В следующих строках заданы значения матрицы по строкам и столбцам. В матрице всегда есть только одна фигура. Фигура отстоит от краев матрицы минимум на один ноль. Начало координат в левом верхнем углу. Координаты растут вниз и вправо.

**Формат выходных данных**

Координаты верхнего левого и правого нижнего угла прямоугольника отделенные пробелами. Координаты задаются номером строки и номером столбца. Нумерация начинается с 0.

Ход решения:

#include <iostream>

#include <vector>

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 minRow = h, maxRow = -1, minCol = w, maxCol = -1;

    for (int i = 0; i < h; ++i) {

        for (int j = 0; j < w; ++j) {

            if (matrix[i][j] == 1) {

                minRow = min(minRow, i);

                maxRow = max(maxRow, i);

                minCol = min(minCol, j);

                maxCol = max(maxCol, j);

            }

        }

    }

    cout << minRow - 1 << " " << minCol - 1 << " " << maxRow + 1 << " " << maxCol + 1 << endl;

    return 0;

}

Результат:

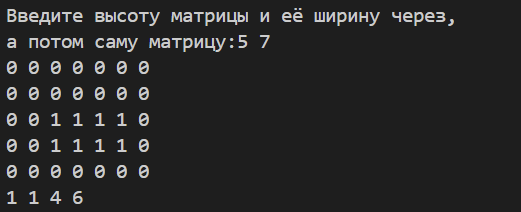


Рисунок 16 – Пример работы

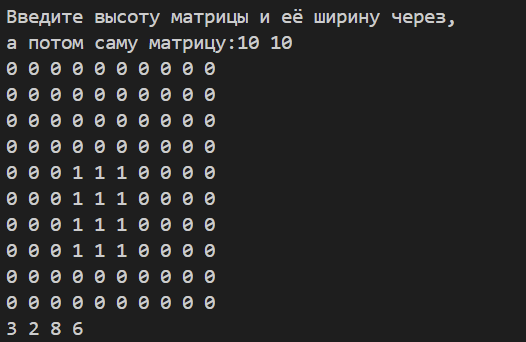


Рисунок 17 – Пример работы

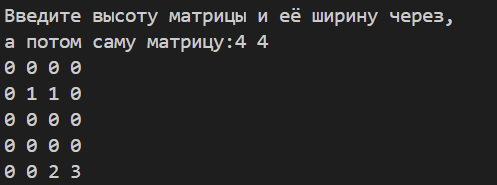


Рисунок 18 – Пример работы

**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.

**Формат входных данных**

В первой строке пишем число n (не больше 50). Далее в n строках пишем по 4 натуральных числа (a, b, x, y).

**Формат выходных данных**

Выводим n чисел через пробел, каждое число – максимальное число работающих пар из микроконтроллера и модуля управления мотором можно составить для строки.

Ход решения:

#include <iostream>

#include <vector>

using namespace std;

int main() {

    int n;

    cout << "введите количество запросов:";

    cin >> n;

    vector<int> results;

    for (int i = 0; i < n; ++i) {

        int a, b, x, y;

        cin >> a >> b >> x >> y;

        int pairs1 = min(a, x);

        int availableA = a - pairs1;

        int availableB = b;

        int pairs2 = min(availableA + availableB, y);

        int totalPairs = pairs1 + pairs2;

        results.push\_back(totalPairs);

    }

    for (int i = 0; i < n; ++i) {

        cout << results[i] << " ";

    }

    cout << endl;

    return 0;

}

Результат:

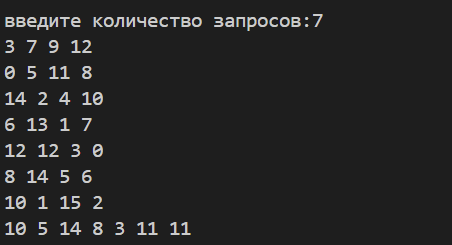


Рисунок 19 – Пример работы

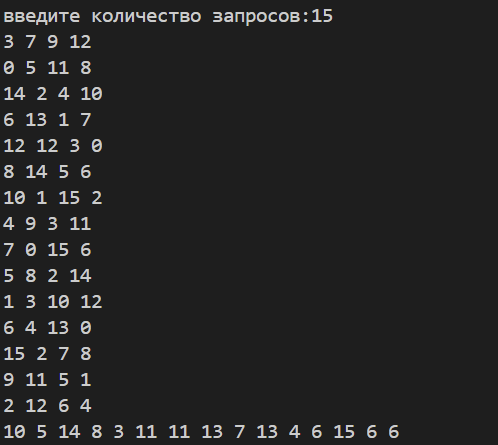


Рисунок 20 – Пример работы

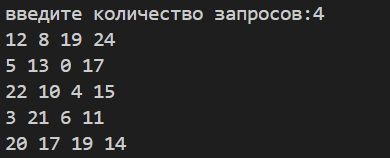


Рисунок 21 – Пример работы

**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.

**Формат входных данных**

В единственной строке находится строка из шести символов, состоящая из цифр и заглавных латинских букв.

**Формат выходных данных**

Если строка является автомобильным номером, то необходимо вывести "Yes", в ином случае – "No" без кавычек.

Ход решения:

#include <iostream>

#include <string>

using namespace std;

bool isValidCarNumber(const string& number) {

    if (number.length() != 6) return false;

    if (!isupper(number[0])) return false;

    for (int i = 1; i <= 3; i++) {

        if (!isdigit(number[i])) return false;

    }

    for (int i = 4; i < 6; i++) {

        if (!isupper(number[i])) return false;

    }

    return true;

}

int main() {

    string carNumber;

    cout << "Введите номер машины: ";

    cin >> carNumber;

    if (isValidCarNumber(carNumber)) {

        cout << "Yes\n";

    } else {

        cout << "No\n";

    }

    return 0;

}

Результат:



Рисунок 22 – Пример работы



Рисунок 23 – Пример работы



Рисунок 24 – Пример работы

**Задание на практику № 8**

Постановка задачи:

Составить светодиодную матрицу размером не менее 8 на 8 светодиодов (пример на рисунке ниже размером 4 на 4)

!Обратите внимание на ориентацию светодиодов на поле



Рисунок 25 – Схема подключения матрицы 4 на 4

На матрицу вывести инфографику с различными динамично меняющимися изображениями.

Ход решения:

#include "Adafruit\_NeoPixel.h"

#define PIN 3

#define NUMPIXELS 64

#define LETTER\_DELAY 2000

#define PIXEL\_DELAY 50

Adafruit\_NeoPixel strip(NUMPIXELS, PIN, NEO\_GRB + NEO\_KHZ800);

const uint32\_t LETTER\_COLOR = strip.Color(0, 0, 255);

const uint32\_t BG\_COLOR = strip.Color(0, 0, 0);

const uint8\_t letters[6][8][8] = {

  {

    {0,0,0,0,0,0,0,0},

    {0,1,0,0,0,0,1,0},

    {0,1,1,0,0,0,1,0},

    {0,1,0,0,1,0,1,0},

    {0,1,0,0,1,0,1,0},

    {0,1,1,0,0,0,1,0},

    {0,1,0,0,0,0,1,0},

    {0,0,0,0,0,0,0,0}

  },

  {

    {0,0,0,0,0,0,0,0},

    {0,1,1,1,1,1,0,0},

    {0,1,0,0,0,0,1,0},

    {0,1,0,0,0,0,1,0},

    {0,0,1,1,1,1,1,0},

    {0,1,0,0,0,0,0,0},

    {0,0,0,0,0,0,1,0},

    {0,0,0,0,0,0,0,0}

  },

  {

    {0,0,0,0,0,0,0,0},

    {0,1,0,0,0,0,1,0},

    {0,1,0,0,0,0,1,0},

    {0,1,1,1,1,1,1,0},

    {0,1,1,1,1,1,1,0},

    {0,1,0,0,0,0,1,0},

    {0,1,0,0,0,0,1,0},

    {0,0,0,0,0,0,0,0}

  },

  {

    {0,0,0,0,0,0,0,0},

    {0,1,0,0,0,0,1,0},

    {0,1,1,0,0,0,1,0},

    {0,1,0,0,1,0,1,0},

    {0,1,0,0,1,0,1,0},

    {0,1,1,0,0,0,1,0},

    {0,1,0,0,0,0,1,0},

    {0,0,0,0,0,0,0,0}

  },

  {

    {0,0,0,0,0,0,0,0},

    {0,1,1,1,1,1,1,0},

    {0,1,1,1,1,1,1,0},

    {0,0,0,1,1,0,0,0},

    {0,0,0,1,1,0,0,0},

    {0,0,0,1,1,0,0,0},

    {0,0,0,1,1,0,0,0},

    {0,0,0,0,0,0,0,0}

  },

  {

    {0,0,0,0,0,0,0,0},

    {0,1,0,0,0,0,1,0},

    {0,1,0,0,0,0,1,0},

    {0,1,1,1,1,1,1,0},

    {0,1,0,0,0,0,0,0},

    {0,0,0,0,0,0,1,0},

    {0,1,1,1,1,1,0,0},

    {0,0,0,0,0,0,0,0}

  }

};

void clearMatrix() {

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

    strip.setPixelColor(i, BG\_COLOR);

  }

  strip.show();

}

void drawLetterWithPixelDelay(uint8\_t letterIndex) {

  clearMatrix();

  int activePixels[64][2];

  int count = 0;

  for(int row = 0; row < 8; row++) {

    for(int col = 0; col < 8; col++) {

      if(letters[letterIndex][row][col]) {

        activePixels[count][0] = row;

        activePixels[count][1] = col;

        count++;

      }

    }

  }

  for(int i = 0; i < count; i++) {

    int row = activePixels[i][0];

    int col = activePixels[i][1];

    strip.setPixelColor(row \* 8 + col, LETTER\_COLOR);

    strip.show();

    delay(PIXEL\_DELAY);

  }

}

void setup() {

  strip.begin();

  strip.setBrightness(100);

  clearMatrix();

  delay(1000);

}

void loop() {

  for(int i = 0; i < 6; i++) {

    drawLetterWithPixelDelay(i);

    delay(LETTER\_DELAY);

    clearMatrix();

    delay(200);

  }

  delay(1000);

}

Результат:

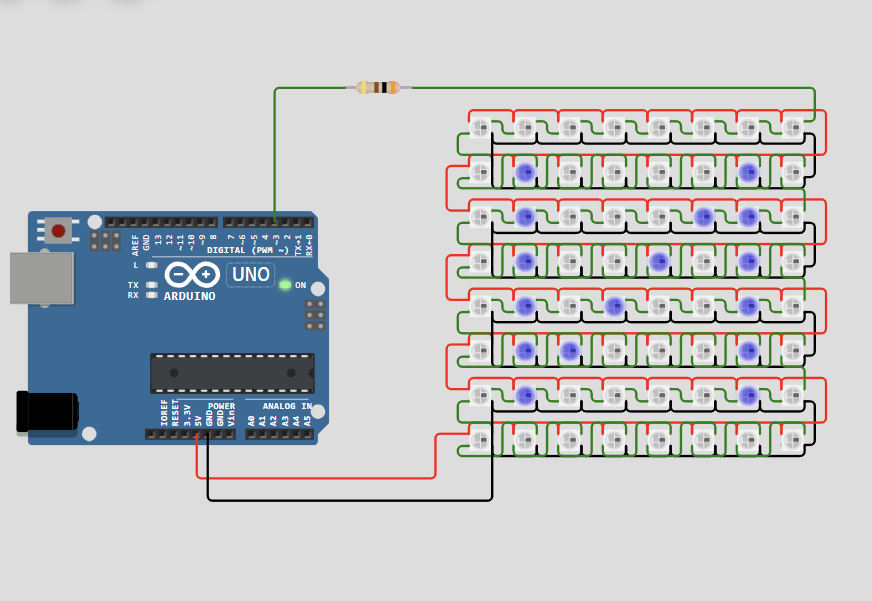


Рисунок 26 – Пример работы

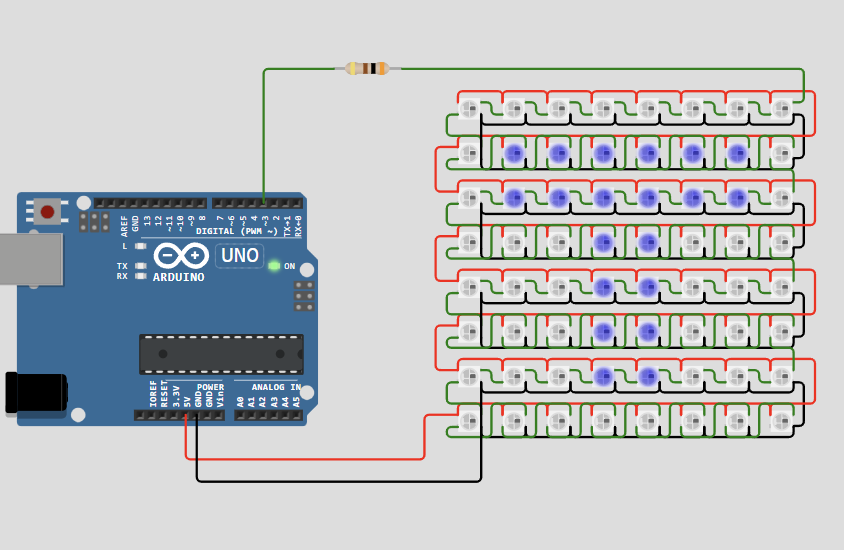


Рисунок 27 – Пример работы

**Задание № 9**

Постановка задачи:

**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.



Рисунок 28 – Схема управления работой автоматических дверей

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



Рисунок 29 – Схема изменение значения фоторезистора

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 redLedPin = 9;

const int greenLedPin = 5;

const int photoResistorPin = A0;

const int threshold = 512;

const unsigned long doorOpenTime = 5000;

bool doorOpen = false;

unsigned long doorOpenStartTime = 0;

void setup() {

  pinMode(redLedPin, OUTPUT);

  pinMode(greenLedPin, OUTPUT);

**Serial**.begin(9600);

  digitalWrite(redLedPin, HIGH);

  digitalWrite(greenLedPin, LOW);

**Serial**.println("Двери закрыты");

}

void loop() {

  int sensorValue = analogRead(photoResistorPin);

  if (sensorValue > threshold) {

    if (!doorOpen) {

      doorOpen = true;

      doorOpenStartTime = millis();

      digitalWrite(redLedPin, LOW);

      digitalWrite(greenLedPin, HIGH);

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

    } else {

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

        sensorValue = analogRead(photoResistorPin);

        if (sensorValue <= threshold) {

          doorOpen = false;

          digitalWrite(greenLedPin, LOW);

          digitalWrite(redLedPin, HIGH);

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

        } else {

          doorOpenStartTime = millis();

        }

      }

    }

  } else if (doorOpen && (millis() - doorOpenStartTime >= doorOpenTime)) {

    doorOpen = false;

    digitalWrite(greenLedPin, LOW);

    digitalWrite(redLedPin, HIGH);

**Serial**.println("Двери закрываются (время истекло)");

  }

  delay(100);

}

Результат:

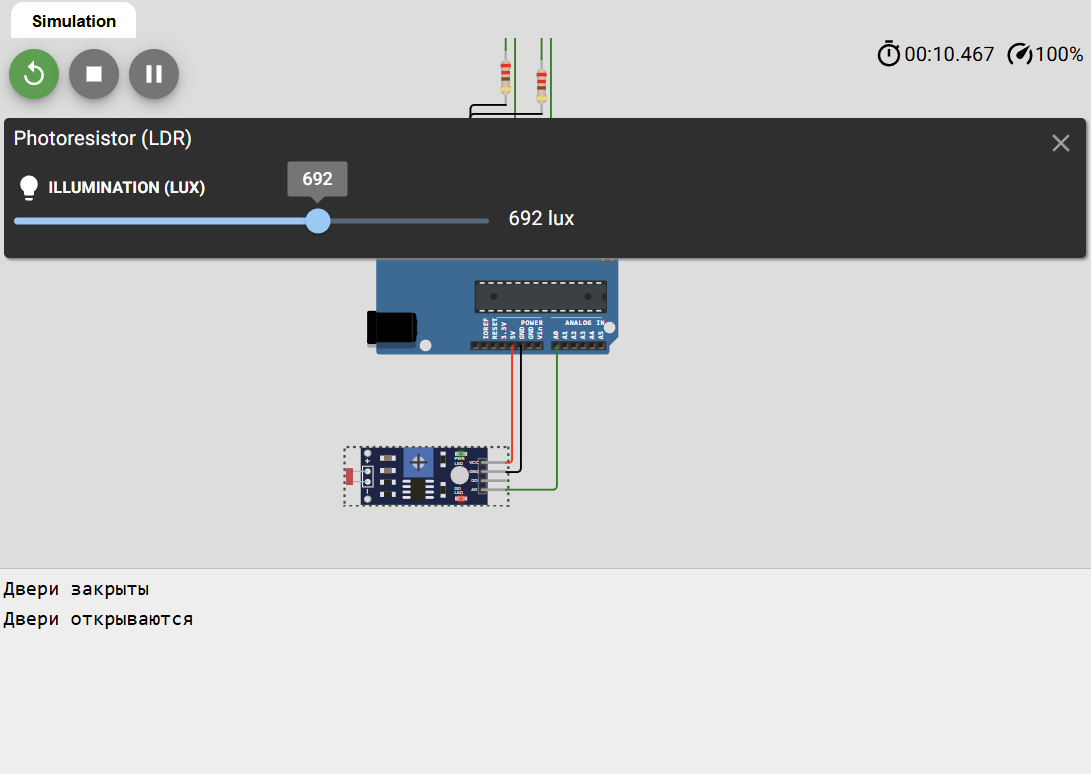


Рисунок 30 – Пример работы

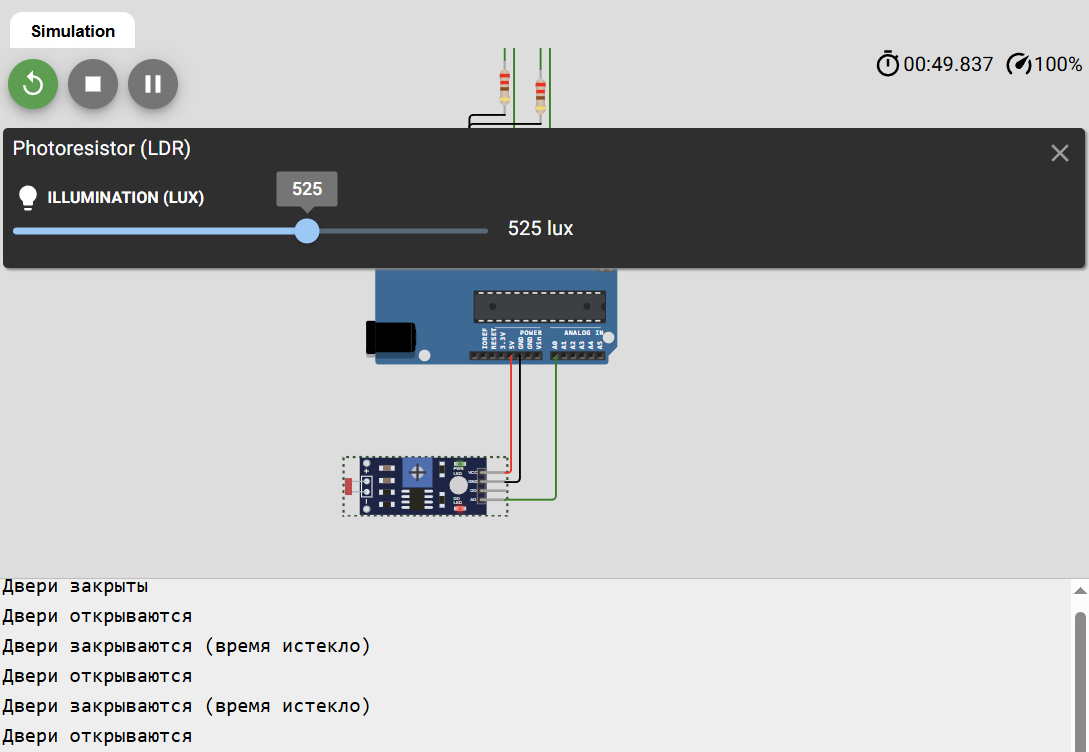


Рисунок 31 – Пример работы

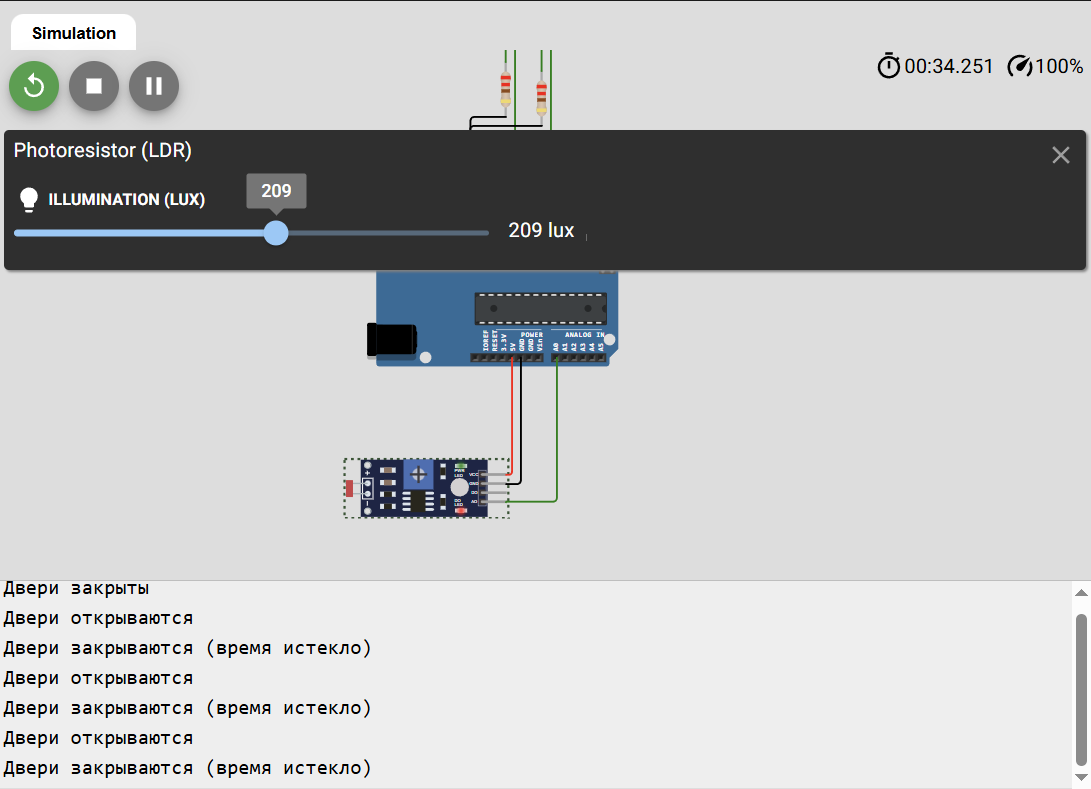


Рисунок 32 – Пример работы

Сылка на схему: <https://wokwi.com/projects/434473120345576449>

**Задание № 10**

Постановка задачи:

**Задачи:**

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



Рисунок 33 – Схема подключение сервопривода

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;

void setup() {

  myservo.attach(3);

**Serial**.begin(9600);

**Serial**.println("Введите угол (0-180):");

  myservo.write(currentAngle);

}

void loop() {

  if (**Serial**.available() > 0) {

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

    input.trim();

    bool isNumber = true;

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

      if (!isdigit(input.charAt(i))) {

        isNumber = false;

        break;

      }

    }

    if (isNumber) {

      targetAngle = input.toInt();

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

**Serial**.print("Поворот на: ");

**Serial**.print(targetAngle);

**Serial**.println("°");

        if (targetAngle != currentAngle) {

          int step = (targetAngle > currentAngle) ? 1 : -1;

          while (currentAngle != targetAngle) {

            currentAngle += step;

            myservo.write(currentAngle);

            delay(15);

          }

        }

      } else {

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

      }

    } else {

**Serial**.println("Ошибка: введите число!");

    }

  }

}

Результат:

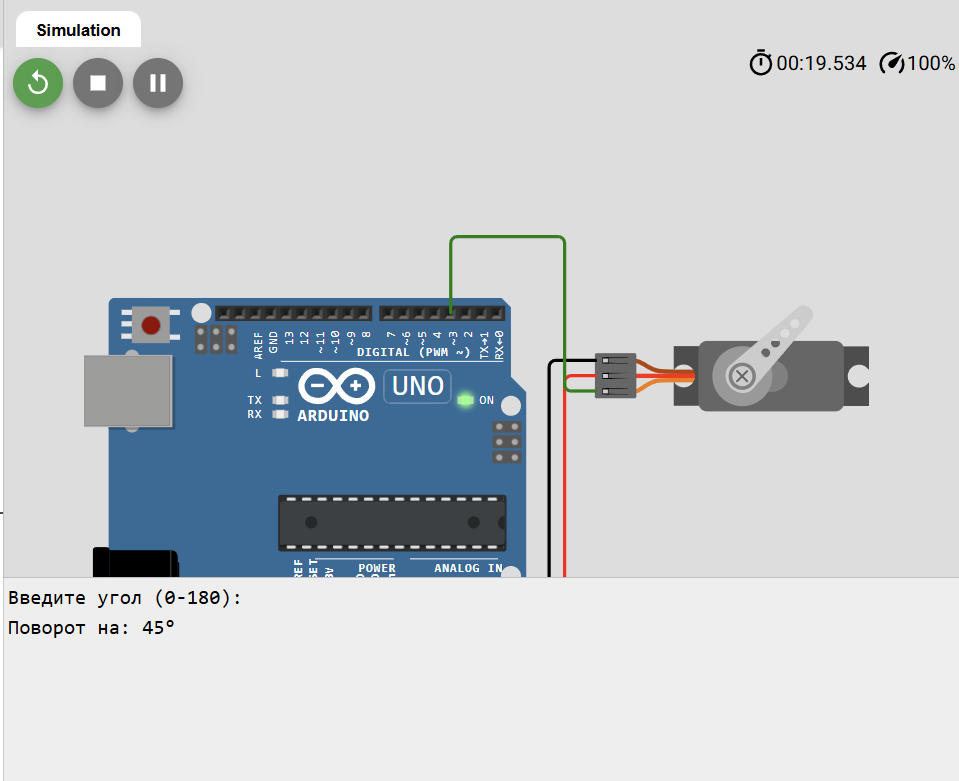


Рисунок 34 – Пример работы

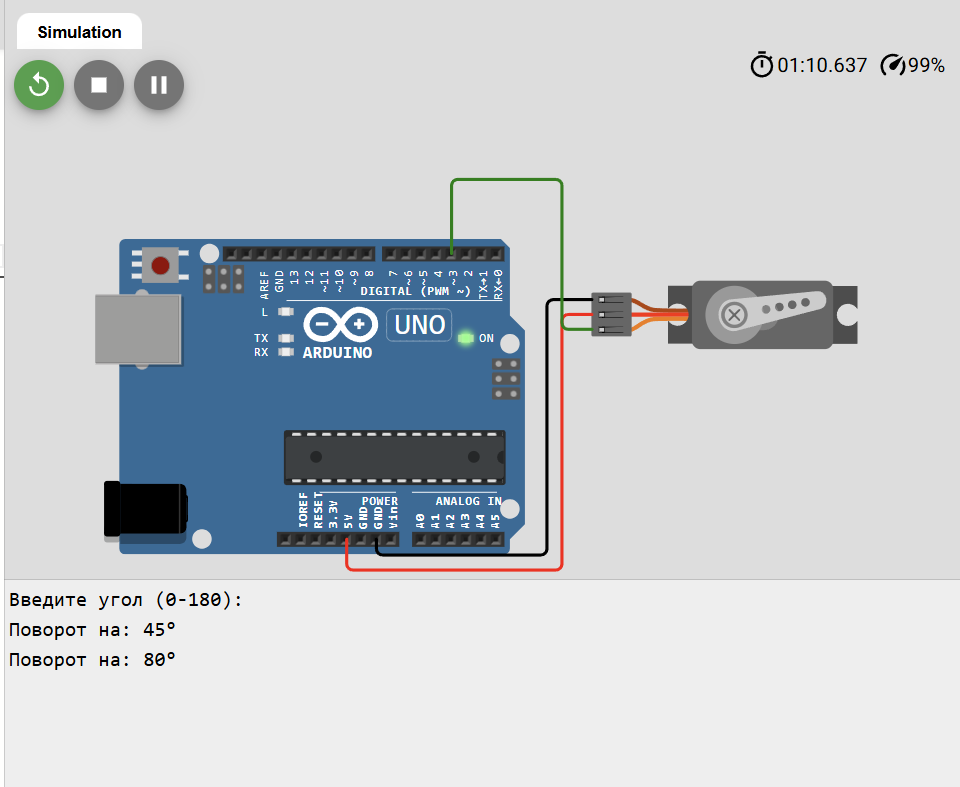


Рисунок 35 – Пример работы

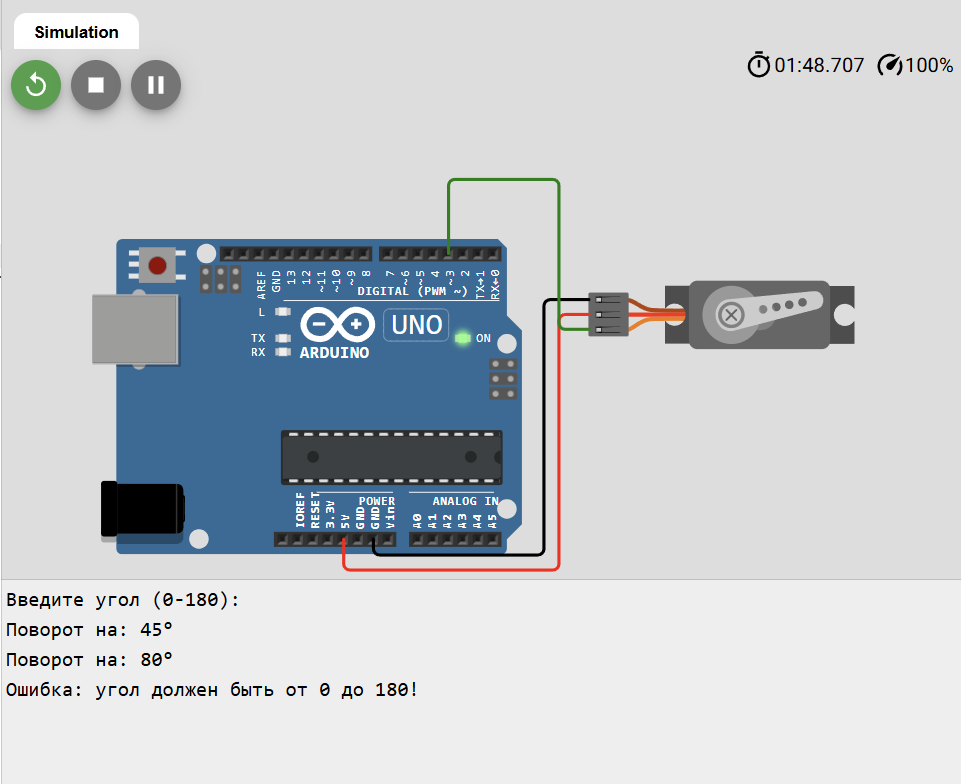


Рисунок 36 – Пример работы

Ссылка на схему: <https://wokwi.com/projects/434474188642265089>