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 Цыдендамбаев Д.А., ИСИБ-24-1

(ФИО, группа, подпись)

[Резюме hh.ru](https://irkutsk.hh.ru/resume/b02b916cff0efe781f0039ed1f356739437373?hhtmFrom=resume_list)

[Резюме superjob](https://www.superjob.ru/resume/programmist-s-55734466.html)

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 попрактике

(ФИО, подпись, дата)

Содержаниеотчетана \_\_\_ стр. Приложение котчетуна \_\_\_ стр.

Иркутск 2025

**Индивидуальное задание на прохождение**

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| для | Цыдендамбаева Дмитрия Александровича | | | | |
|  | (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» июня2025 г. по «29» июня 2025г.

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 | | | | | Цыдендамбаева Дмитрия Александровича,  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 подразделенияназначен:

Кононенко Роман Владимирович, доцент института ИТиАД

(ФИО, должность)

**Рабочий график (план) прохожденияпрактической подготовки**

(заполняется обучающимся)

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Период  практики | Содержание выполненных работ | Подпись руководителя практики от структурного подразделения |
| 1 | 16.06.2025 | Решил задачу №1,  Изучил теоретический материал для задачи №2,  Изучил теоретический материал для задачи №3. |  |
| 2 | 17.06.2025 | Решил задачу №2.  Решил задачу №3. |  |
| 3 | 18.06.2025 | Решил задачу №4. |  |
| 4 | 19.06.2025 | Решил задачу №5. |  |
| 5 | 21.06.2025 | Решил задачу №6.  Решил задачу №6. Решил задачу №8. |  |
| 6 | 26.06.2025 | Решил задачу №9.  Решил задачу №10. Решил задачу №11. |  |
| 7 |  |  |  |
| 8 |  |  |  |
| 9 |  |  |  |
| 10 |  |  |  |

|  |  |
| --- | --- |
| Дата фактического прибытия |  |
| обучающегося вструктурное 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) |

**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).

**Код программы:**

#include <iostream>

#include <vector>

void buildHeap(int n) {

std::vector<std::vector<int>> heap;

std::vector<std::vector<int>> original;

int value = 1;

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

std::vector<int> row, copyRow;

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

if (j <= i) {

row.push\_back(value \* 10 + i + j); // генерируем

copyRow.push\_back(value \* 10 + i + j);

value++;

}

else {

row.push\_back(0);

copyRow.push\_back(0);

}

}

heap.push\_back(row);

original.push\_back(copyRow);

}

// пирамида

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

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

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

}

std::cout << std::endl;

}

// дп

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

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

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

}

}

std::vector<int> path;

int i = 0, j = 0;

path.push\_back(original[i][j]);

while (i < n - 1) {

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

++i;

}

else {

++i;

++j;

}

path.push\_back(original[i][j]);

}

std::cout << "\nPath: ";

for (size\_t k = 0; k < path.size(); ++k) {

std::cout << path[k];

if (k != path.size() - 1)

std::cout << " ";

}

std::cout << std::endl;

std::cout << "Min time: " << heap[0][0] << std::endl;

}

int main() {

int n = 0;

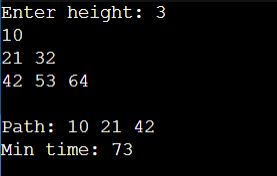
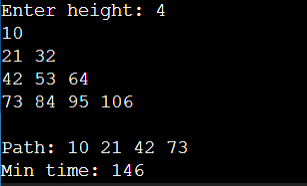
std::cout << "Enter height: ";

std::cin >> n;

buildHeap(n);

return 0;

}



**Задача №2**

После метеоритной атаки компьютерная сеть для управления лунными заводами разбилась на части, нужно объединить её в единое целое. Каждый фрагмент сети представлен в виде ненаправленного графа.

Вам известно общее число вершин графа (узлы сети, не более 1000) и набор рёбер (сохранившиеся линии связи, не более 1000).

Определите, какое минимальное число линий связи нужно дополнительно построить, чтобы сеть стала единой.

**Код программы:**

**#include <iostream>**

**#include <vector>**

**void dfs(int v, const std::vector<std::vector<int>>& g, std::vector<bool>& visited) {**

**visited[v] = true;**

**for (int u : g[v]) {**

**if (!visited[u]) {**

**dfs(u, g, visited);**

**}**

**}**

**}**

**int main() {**

**int n, m;**

**std::cin >> n >> m;**

**std::vector<std::vector<int>> g(n);**

**for (int i = 0; i < m; ++i) {**

**int a, b;**

**std::cin >> a >> b;**

**--a; --b; // делаем индексацию с нуля**

**g[a].push\_back(b);**

**g[b].push\_back(a);**

**}**

**std::vector<bool> visited(n, false);**

**int components = 0;**

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

**if (!visited[i]) {**

**dfs(i, g, visited);**

**components++;**

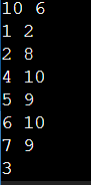
**}**

**}**

**std::cout << (components - 1) << std::endl;**

**return 0;**

**}**



**Задача 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.

#include <iostream>

#include <vector>

int main() {

int n, k;

std::cin >> n >> k;

std::vector<long long> streets(n + 1, 0); // Индексы улиц от 1 до n

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

int type;

std::cin >> type;

if (type == 1) {

int ix;

long long x;

std::cin >> ix >> x;

streets[ix] += x;

} else if (type == 2) {

int u, r;

std::cin >> u >> r;

long long sum = 0;

for (int j = u; j <= r; ++j) {

sum += streets[j];

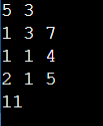
}

std::cout << sum << endl;

}

}

return 0;   
}



**Задача 4**

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

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

**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.**

#include <iostream>

#include <vector>

#include <iostream>

#include <vector>

int main() {

int n, k;

std::cin >> n >> k; // ввод

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

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

std::cin >> permutation[i]; // читаем перестановку

}

std::string encrypted;

std::cin >> encrypted;

// Обратная перестановка

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

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

inverse[permutation[i] - 1] = i + 1;

}

std::string result = encrypted;

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

std::string temp = result;

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

temp[i] = result[inverse[i] - 1]; // переставляем

}

result = temp;

}

std::cout << result; // результат

return 0;

}  
  
**Задача 5**

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

**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.**#include <iostream>

#include <vector>

#include <climits>

int main() {

int h, w;

std::cin >> h >> w; // размерность матрицы

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

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

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

std::cin >> matrix[i][j];

}

}

int min\_row = INT\_MAX, max\_row = -1;

int min\_col = INT\_MAX, max\_col = -1;

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

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

if (matrix[i][j] == 1) { // если часть фигуры

if (i < min\_row) min\_row = i;

if (i > max\_row) max\_row = i;

if (j < min\_col) min\_col = j;

if (j > max\_col) max\_col = j;

}

}

}

// выводим координаты

std::cout << min\_row << " " << min\_col << " ";

std::cout << max\_row << " " << max\_col << std::endl;

return 0;

}

 **Задача 6**В школьном кружке робототехники есть два вида микроконтроллеров (условно тип A и тип B) и два вида модулей управления мотором (условно тип 1 и тип 2). Выяснилось, что контроллер типа B и модуль управления типа 2 несовместимы. Использование микроконтроллеров и модулей управления в других комбинациях возможно. Имеется a микроконтроллеров типа A, b микроконтроллеров типа B, x модулей управления типа 1 и y модулей типа 2. Определите, какое максимальное число работающих пар из микроконтроллера и модуля управления мотором можно составить. Ваша программа должна ответить на n запросов.

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.  
  
  
 #include <iostream>

#include <vector>

#include <algorithm>

int main() {

int n = 0;

std::cout << "Введите количество запросов: ";

std::cin >> n; // читаем количество запросов

std::vector<std::vector<int>> requests(n, std::vector<int>(4)); // массив для хранения всех запросов

std::cout << "Введите запросы (a, b, x ,y): ";

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

for (int j = 0; j < 4; j++) {

std::cin >> requests[i][j]; // считываем a, b, x, y для каждого запроса

}

}

std::cout << "Результат: ";

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

std::vector<int> req = requests[i]; // текущий запрос

int a = req[0]; // количество контроллеров a

int b = req[1]; // количество контроллеров b

int x = req[2]; // количество модулей типа 1

int y = req[3]; // количество модулей типа 2

int used\_b = 0;

// сначала подключаем b к модулю типа 1 (совместимость только с ними)

if (b >= x) {

used\_b = x; // b хватает на все x

x = 0; // все модули типа 1 заняты

} else {

used\_b = b; // используем все b

x -= b; // оставшиеся модули типа 1

}

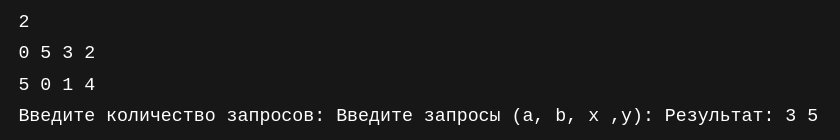
// затем подключаем a к оставшимся модулям типа 1 и ко всем модулям типа 2

int used\_a = std::min(a, x + y);

std::cout << used\_b + used\_a << ' ';

}

return 0;

**Задача 7**На компьютере работника автосервиса нашли файл с последовательностью автомобильных номеров, обслуживавшихся в этом автосервисе. Так как файл был поврежден, некоторые данные отображаются неверно. Нужно определить, какие из них остались невредимыми.

Автомобильным номером является строка из шести символов. Первый символ – заглавная латинская буква, далее следует 3 цифры, и после – две заглавные латинские буквы. Например, строка "P142EQ" является номером. Вам будет дана строка, состоящая из шести символов, необходимо ответить, является ли строка автомобильным номером.

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.  
  
#include <iostream>

#include <cctype>

int main() {

std::string plate;

std::cout << "введите строку из 6 символов: ";

std::cin >> plate; // читаем ввод

bool format\_ok = true; // предполагаем, что всё верно

// проверяем первую букву

if (!std::isalpha(plate[0])) format\_ok = false;

// проверяем три цифры

if (!std::isdigit(plate[1]) || !std::isdigit(plate[2]) || !std::isdigit(plate[3])) format\_ok = false;

// проверяем две последние буквы

if (!std::isalpha(plate[4]) || !std::isalpha(plate[5])) format\_ok = false;

std::cout << "результат: " << (format\_ok ? "Yes" : "No");

return 0;

}  
