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 Храмцова Никиты Владимировича, ИСИб-24-1

(ФИО, группа, подпись)

Руководитель практики от института ИТиАД

Кононенко Р. В., Доцент института ИТиАД

(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 \_\_\_ стр. Приложение к отчету на \_\_\_ стр.

Иркутск 2025

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

**Индивидуальное задание на прохождение**

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| для | Храмцова Никиты Владимировича | | | | |
|  | (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 «15» сентября 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

\_\_\_\_\_\_\_\_\_\_«16» Июня 2025 г.

(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

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 <algorithm>

#include <cstdlib>

#include <ctime>

int main() {

srand(time(0));

int N;

std::cout << "Введите высоту пирамиды (N): ";

std::cin >> N;

std::vector<std::vector<int>> pyramid(N, std::vector<int>(N));

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

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

pyramid[i][j] = rand() % 1000 + 1;

}

}

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

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

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

}

std::cout << std::endl;

}

// Динамическое программирование (снизу вверх)

std::vector<std::vector<int>> dp = pyramid;

for (int i = N - 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]);

}

}

// Восстановление пути

std::vector<int> path;

int current\_j = 0;

path.push\_back(pyramid[0][current\_j]);

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

if (dp[i][current\_j] < dp[i][current\_j + 1]) {

}

else {

current\_j++;

}

path.push\_back(pyramid[i][current\_j]);

}

std::cout << "\nМинимальная сумма: " << dp[0][0] << std::endl;

std::cout << "Путь: ";

for (int num : path) {

std::cout << num << " ";

}

std::cout << std::endl;

}

Результаты

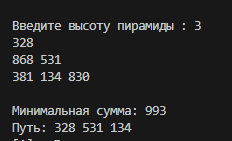


Рисунок 2

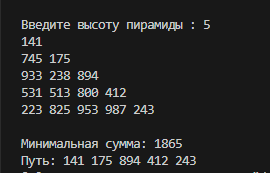


Рисунок 3

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 4



Рисунок 5

**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>

enum Colors { White, Grey, Black };

class Graph {

private:

std::vector<std::vector<int>> vertex\_;

std::vector<Colors> colors\_;

std::vector<int> parents\_;

public:

Graph(int n, int m) {

vertex\_.resize(n);

colors\_.resize(n, White);

parents\_.resize(n, -1);

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

int v1 = 0;

int v2 = 0;

std::cin >> v1 >> v2;

vertex\_[--v1].push\_back(--v2);

vertex\_[v2].push\_back(v1);

}

}

void DFS(int index) {

colors\_[index] = Grey;

for (int children : vertex\_[index]) {

if (colors\_[children] == White) {

parents\_[children] = index;

DFS(children);

}

}

colors\_[index] = Black;

}

void PrintColors() {

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

std::cout << colors\_[i] << ' ';

}

}

int NumComponents() {

int count = 0;

DFS(0);

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

if (colors\_[i] == White) {

count++;

DFS(i);

}

}

return count;

}

};

int main() {

int n = 0;

int m = 0;

std::cin >> n >> m;

Graph graph(n, m);

std::cout << "Минимальное число дополнительных ребер: " << graph.NumComponents() << ' ';

}

Результат

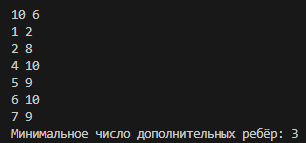


Рисунок 6

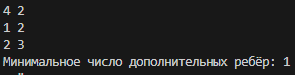


Рисунок 7

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>

int main() {

int n = 0;

int m = 0;

std::cin >> n >> m;

int\* roads = new int[n + 1];

int\* answers = new int[m];

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

answers[i] = -1;

}

int count = 0;

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

int id = 0;

int first\_num = 0;

int second\_num = 0;

std::cin >> id >> first\_num >> second\_num;

if (id == 1) {

roads[first\_num] = second\_num;

}

else if (id == 2) {

int sum = 0;

for (int j = first\_num; j <= second\_num; j++) {

sum += roads[j];

}

answers[count++] = sum;

}

}

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

if (answers[i] != -1) {

std::cout << answers[i] << '\n';

}

}

delete[] roads;

delete[] answers;

}

Результат

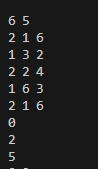


Рисунок 8



Рисунок 9

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 10

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 <string>

void BubbleSort(int\* arr, std::string& arr2, int size) {

for (int i = 0; i < size - 1; i++) {

for (int j = 0; j < size - i - 1; j++) {

if (arr[j] > arr[j + 1]) {

std::swap(arr[j], arr[j + 1]);

std::swap(arr2[j], arr2[j + 1]);

}

}

}

}

int main() {

int n = 0;

int k = 0;

std::cin >> n >> k;

std::string words;

int\* arr = new int[n];

int\* arr\_copy = new int[n];

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

std::cin >> arr[i];

}

std::cin >> words;

std::copy(arr, arr + n, arr\_copy);

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

BubbleSort(arr, words, n);

std::copy(arr\_copy, arr\_copy + n, arr);

}

std::cout << words;

delete[] arr;

delete[] arr\_copy;

}

Результат

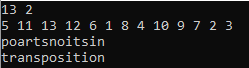


Рисунок 11



Рисунок 12

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>

int main() {

int n = 0;

std::cin >> n;

int a, b, x, y, sum = 0;

int\* result = new int[n];

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

sum = 0;

a, b, x, y = 0;

std::cin >> a >> b >> x >> y;

if (b - x >= 0) {

sum += x;

x = 0;

}

else {

sum += b;

x = x - b;

}

if (a - x > 0) {

sum += x;

a = a - x;

if (a - y >= 0) {

sum += y;

y = 0;

}

else {

sum += a;

y = y - a;

}

}

else {

sum += a;

y = y - a;

}

result[i] = sum;

}

std::cout << "\nResult\n";

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

std::cout << result[i] << ' ';

}

delete[] result;

}

Результат

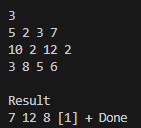


Рисунок 13

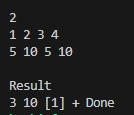


Рисунок 14

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>

void CheckPlate(std::string plate) {

if (plate.size() != 6) {

std::cout << "No";

return;

}

if (plate[0] >= 'A' && plate[0] <= 'Z') {

for (int i = 1; i < 4; i++) {

if (plate[i] < '0' || plate[i] > '9') {

std::cout << "No";

return;

}

}

if ((plate[4] >= 'A' && plate[4] <= 'Z') && (plate[5] >= 'A' && plate[5] <= 'Z')) {

std::cout << "Yes";

} else {

std::cout << "No";

}

} else{

std::cout << "No";

}

}

int main() {

std::string plate;

std::cin >> plate;

CheckPlate(plate);

}

Результат



Рисунок 15



Рисунок 16