МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации



**ОТЧЁТ**

**по практической работе №3**

**«**Порядок сборки проекта на языке C++**»**

**по дисциплине: «***Программирование***»**

Выполнил:Проверил:

Студент гр. «АБс-324», «АВТФ» *доцент кафедры ЗИ*

*Петров М.И. Архипова А. Б.*

«\_\_» \_\_\_\_\_\_ 2024 г.«\_\_\_» \_\_\_\_\_\_ 2024 г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (подпись)

Новосибирск 2024

**Цели и задачи работы**: написать 10 математических задач которые решаются с помощью алгоритма и итератора.

**Задача 1:** В НГТУ есть 6 корпусов, связанные друг с другом переходами. Нужно найти оптимальный маршрут от 1-ого корпуса до 6-ого корпуса, учитывая необходимость посещения определенных промежуточных корпусов.

**Задача 2:** В одном городе работают две транспортные компании, предоставляющие услуги по грузоперевозкам. Первая компания осуществляет доставку посылок маленького объема, а вторая занимается перевозками крупных грузов. Компании решили сотрудничать, чтобы предоставлять полные услуги. Для этого им необходимо объединить базы данных грузов. В каждой базе есть отдельные вектора с описанием грузов. Необходимо создать новый общий вектор с описанием всех грузов, отсортированный по размеру и весу.

**Задача 3:** Во время проведения олимпиады каждый из участников получил свой идентификационный номер. Необходимо отсортировать список участников олимпиады по количеству набранных ими баллов. Необходимо вывести исходный список в порядке убывания баллов. Если у некоторых участников одинаковые баллы, то их нужно отсортировать в порядке возрастания идентификационного номера.

**Задача 4:** В деканате проводится проверка посещаемости учащихся на экзамене по математическому анализу. Дано: N - количество студентов, явившихся на экзамен; строка, содержащая информацию о каждом студенте: его фамилию, группу и количество баллов, набранных на экзамене; а также минимальное количество баллов для сдачи экзамена. Требуется определить, кто из студентов сдал экзамен успешно, а кто остался на пересдачу, не набрав минимальный балл.

**Задача 5:** Есть набор данных, представляющих собой список студентов вуза, у каждого из которых есть свой уникальный идентификационный номер, группа и данные о том получает ли он ежемесячные выплаты или нет. Нужно написать итератор, который позволит получить список всех студентов определенной группы с информацией о том получает ли он выплаты или нет.

**Задача 6:** Студент носит с собой набор из N предметов, каждый из которых имеет определенную стоимость и вес. Также есть рюкзак определенной вместимости. Необходимо выбрать такой поднабор предметов, чтобы их суммарный вес не превышал вместимость рюкзака, а суммарная стоимость была максимальной.

**Задача 7:** На дне птичьего гнезда находилось N птенцов разного цвета. Всем этим птенцам нужно расставить номера по убыванию их размеров. Одинаковые размеры обозначают одинаковые номера, которые нужно расставить по возрастанию номеров по цвету птенцов. Красный птенец никогда не может иметь номер менее 3. Птенцу с номером 4 нравится иметь по номеру разный цвет. В том случае, если у двух птенцов совпадают как цвет, так и размер, первым в списке выписывается тот птенец, у которого номер меньше.

**Задача 8:** Дана последовательность чисел. Необходимо написать программу, которая будет находить среднее арифметическое всех чисел в этой последовательности с использованием итераторов. Пользователь вводит числа последовательно, после чего вводит команду "stop", чтобы завершить ввод.

**Задача 9:** Дан массив чисел. Необходимо реализовать алгоритм, который найдет среднее арифметическое только тех чисел, которые расположены на четных позициях в массиве.

**Задача 10:** Дано целое число N. Необходимо найти сумму квадратов четных чисел в диапазоне от 1 до N.

Задание 1

**Решение на C++**

#include <iostream>

#include <vector>

#include <queue>

using namespace std;

const int INF = 1e9;

const int numBuildings = 6;

vector<vector<pair<int, int>>> adjList{

    {},                             // Корпус 0

    {{2, 5}, {3, 7}},               // Корпус 1

    {{1, 5}, {3, 8}, {4, 9}},       // Корпус 2

    {{1, 7}, {2, 8}, {4, 6}},       // Корпус 3

    {{2, 9}, {3, 6}, {5, 7}},       // Корпус 4

    {{4, 7}},                       // Корпус 5

};

vector<int> dijkstra(int start) {

    vector<int> dist(numBuildings + 1, INF);

    dist[start] = 0;

    priority\_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;

    pq.push({0, start});

    while (!pq.empty()) {

        int u = pq.top().second;

        int d = pq.top().first;

        pq.pop();

        if (d > dist[u]) continue;

        for (auto edge : adjList[u]) {

            int v = edge.first;

            int w = edge.second;

            if (dist[u] + w < dist[v]) {

                dist[v] = dist[u] + w;

                pq.push({dist[v], v});

            }

        }

    }

    return dist;

}

int main() {

    vector<int> intermediateBuildings = {2, 4}; // Промежуточные корпуса

    int startBuilding = 1;

    int endBuilding = 6;

    vector<int> startDist = dijkstra(startBuilding);

    vector<int> endDist = dijkstra(endBuilding);

    int shortestPath = INF;

    int intermediateBuilding = 0;

    for (int i : intermediateBuildings) {

        if (startDist[i] + endDist[i] < shortestPath) {

            shortestPath = startDist[i] + endDist[i];

            intermediateBuilding = i;

        }

    }

    cout << "Оптимальный маршрут: Корпус " << startBuilding << " -> Корпус " << intermediateBuilding << " -> Корпус " << endBuilding << endl;

    cout << "Длина кратчайшего пути: " << shortestPath << endl;

    return 0;

}

Задание 2

**Решение на C++**

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

// Структура, описывающая груз

struct Cargo {

    string name; // Название груза

    int size; // Размер груза

    int weight; // Вес груза

};

// Функция сравнения грузов по размеру и весу

bool compareCargo(const Cargo& c1, const Cargo& c2) {

    if (c1.size != c2.size) {

        return c1.size < c2.size; // Сортировка по размеру

    } else {

        return c1.weight < c2.weight; // Сортировка по весу

    }

}

int main() {

    system ("chcp 65001");

    // База данных грузов первой компании (маленькие грузы)

    vector<Cargo> smallCargo = {

        {"Посылка 1", 1, 2},

        {"Посылка 2", 2, 3},

        {"Посылка 3", 3, 1}

    };

    // База данных грузов второй компании (большие грузы)

    vector<Cargo> largeCargo = {

        {"Груз 1", 5, 10},

        {"Груз 2", 4, 8},

        {"Груз 3", 6, 12}

    };

    // Создание общего вектора грузов

    vector<Cargo> allCargo(smallCargo.size() + largeCargo.size());

    // Копирование грузов из базы первой компании

    copy(smallCargo.begin(), smallCargo.end(), allCargo.begin());

    // Копирование грузов из базы второй компании

    copy(largeCargo.begin(), largeCargo.end(), allCargo.begin() + smallCargo.size());

    // Сортировка общего вектора по размеру и весу

    sort(allCargo.begin(), allCargo.end(), compareCargo);

    // Вывод объединенной базы данных грузов

    cout << "Объединенная база данных грузов:\n";

    for (const Cargo& cargo : allCargo) {

        cout << "Название: " << cargo.name << ", Размер: " << cargo.size << ", Вес: " << cargo.weight << endl;

    }

    return 0;

}

Задание 3

**Решение на C++**

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

// Структура для хранения информации об участнике

struct Participant {

    int id; // Идентификационный номер

    int score; // Количество баллов

};

// Функция сравнения участников по количеству баллов

bool compareParticipants(const Participant& p1, const Participant& p2) {

    // Сначала сравниваем по баллам в порядке убывания

    if (p1.score != p2.score) {

        return p1.score > p2.score;

    } else {

        // Если баллы одинаковые, то сравниваем по идентификационному номеру в порядке возрастания

        return p1.id < p2.id;

    }

}

int main() {

    // Список участников олимпиады

    vector<Participant> participants = {{10, 85}, {5, 90}, {3, 80}, {15, 90}, {7, 75}};

    // Сортировка списка участников по количеству баллов

    sort(participants.begin(), participants.end(), compareParticipants);

    // Вывод отсортированного списка

    cout << "Отсортированный список участников:\n";

    for (const Participant& participant : participants) {

        cout << "ID: " << participant.id << ", Баллы: " << participant.score << endl;

    }

    return 0;

}

Задание 4

**Решение на C++**

#include <iostream>

#include <vector>

#include <string>

#include <sstream>

using namespace std;

struct Student {

    string surname;

    string group;

    int score;

};

int main() {

    int N;

    cout << "Введите количество студентов: ";

    cin >> N;

    cin.ignore(); // Удаляем символ новой строки из буфера ввода

    vector<Student> students(N);

    int minScore;

    cout << "Введите минимальное количество баллов для сдачи экзамена: ";

    cin >> minScore;

    cin.ignore(); // Удаляем символ новой строки из буфера ввода

    cout << "Введите информацию о каждом студенте (фамилия, группа, баллы):\n";

    for (int i = 0; i < N; ++i) {

        string line;

        getline(cin, line);

        stringstream ss(line);

        ss >> students[i].surname >> students[i].group >> students[i].score;

    }

    cout << "\nРезультаты экзамена по математическому анализу:\n";

    for (const Student& student : students) {

        if (student.score >= minScore) {

            cout << student.surname << " (" << student.group << ") - сдал экзамен успешно.\n";

        } else {

            cout << student.surname << " (" << student.group << ") - остался на пересдачу.\n";

        }

    }

    return 0;

}

Задание 5

**Решение на C++**

#include <iostream>

#include <vector>

#include <string>

using namespace std;

// Структура для хранения данных о студенте

struct Student {

    int id;

    string group;

    bool receivesPayments;

};

// Итератор для фильтрации студентов по группе и наличию выплат

class StudentsIterator {

public:

    StudentsIterator(vector<Student>& students, const string& group, bool receivesPayments)

        : students(students), group(group), receivesPayments(receivesPayments) {

        current = students.begin();

        findNext();

    }

    Student\* next() {

        Student\* result = &(\*current);

        findNext();

        return result;

    }

    bool hasNext() {

        return current != students.end();

    }

private:

    vector<Student>& students;

    string group;

    bool receivesPayments;

    vector<Student>::iterator current;

    void findNext() {

        while (current != students.end()) {

            if (current->group == group && current->receivesPayments == receivesPayments) {

                break;

            }

            ++current;

        }

    }

};

int main() {

    // Пример использования итератора

    vector<Student> students = {

        {5001, "АБс324", true},

        {3001, "АБс323", false},

        {1001, "АБс323", false},

        {5002, "АБс324", true},

        {4001, "АБс324", true}

    };

    StudentsIterator it(students, "АБс324", true);

    while (it.hasNext()) {

        Student\* student = it.next();

        cout << "ID студента: " << student->id << ", Группа: " << student->group << ", Данные о выплатах: " << boolalpha << student->receivesPayments << endl;

    }

    return 0;

}

Задание 6

**Решение на C++**

#include <iostream>

#include <vector>

using namespace std;

struct Item {

    int weight;

    int value;

};

int knapsack(int capacity, vector<Item>& items) {

    int n = items.size();

    vector<vector<int>> dp(n + 1, vector<int>(capacity + 1, 0));

    for (int i = 1; i <= n; ++i) {

        for (int w = 1; w <= capacity; ++w) {

            if (items[i - 1].weight > w) {

                dp[i][w] = dp[i - 1][w];

            } else {

                dp[i][w] = max(dp[i - 1][w], items[i - 1].value + dp[i - 1][w - items[i - 1].weight]);

            }

        }

    }

    return dp[n][capacity];

}

int main() {

    int capacity = 10;

    vector<Item> items = {{2, 3}, {3, 4}, {4, 5}, {5, 8}};

    cout << "Максимальная стоимость: " << knapsack(capacity, items) << endl;

    return 0;

}

Задание 7

**Решение на C++**

#include <iostream>

#include <map>

#include <vector>

#include <algorithm>

using namespace std;

struct Chick {

    string color;

    int size;

};

bool compareChicks(Chick a, Chick b) {

    if (a.size == b.size) {

        return a.color < b.color;

    }

    return a.size > b.size;

}

int main() {

    int n;

    cout << "Введите количество птенцов: ";

    cin >> n;

    vector<Chick> chicks(n);

    for (int i = 0; i < n; ++i) {

        cout << "Введите цвет и размер птенца " << i+1 << ": ";

        cin >> chicks[i].color >> chicks[i].size;

    }

    sort(chicks.begin(), chicks.end(), compareChicks);

    map<int, int> colorsToNumber;

    int number = 1;

    for (int i = 0; i < n; ++i) {

        if (chicks[i].color == "красный" && number < 3) {

            number = 3;

        }

        colorsToNumber[i] = number;

        ++number;

    }

    for (int i = 0; i < n; ++i) {

        cout << "Птенец " << i+1 << " (" << chicks[i].color << ", " << chicks[i].size << ") имеет номер " << colorsToNumber[i] << endl;

    }

    return 0;

}

Задание 8

**Решение на С++**

#include <iostream>

#include <vector>

#include <iterator>

#include <algorithm>

using namespace std;

int main() {

    vector<int> numbers;

    int num;

    while (true) {

        cout << "Введите число (или 'stop' для завершения ввода): ";

        string input;

        cin >> input;

        if (input == "stop") {

            break;

        } else {

            num = stoi(input);

            numbers.push\_back(num);

        }

    }

    if (!numbers.empty()) {

        double sum = 0.0;

        ostream\_iterator<int> out\_it(cout, " ");

        copy(numbers.begin(), numbers.end(), out\_it);

        sum = accumulate(numbers.begin(), numbers.end(), 0.0);

        double average = sum / numbers.size();

        std::cout << "\nСреднее арифметическое: " << average << std::endl;

    } else {

        std::cout << "Последовательность пуста." << std::endl;

    }

    return 0;

}

Задание 9

**Решение на C++**

#include <iostream>

#include <vector>

using namespace std;

double averageEvenPositions(const vector<int>& arr) {

    int sum = 0;

    int count = 0;

    // Перебираем массив с шагом 2, чтобы получить только элементы на четных позициях

    for (size\_t i = 1; i < arr.size(); i += 2) {

        sum += arr[i];

        count++;

    }

    // Возвращаем среднее арифметическое, если есть элементы на четных позициях

    if (count > 0) {

        return (double)sum / count;

    } else {

        return 0; // Возвращаем 0, если элементов на четных позициях нет

    }

}

int main() {

    vector<int> numbers = {1, 2, 3, 4, 5, 6, 7, 8, 9};

    double average = averageEvenPositions(numbers);

    cout << "Среднее арифметическое чисел на четных позициях: " << average << endl;

    return 0;

}

Задание 10

**Решение на C++**

#include <iostream>

#include <iterator>

using namespace std;

// Итератор для четных чисел

class EvenNumberIterator : public iterator<input\_iterator\_tag, int> {

private:

    int current;

    int limit;

public:

    EvenNumberIterator(int start, int \_limit) : current(start), limit(\_limit) {}

    // Операция сравнения на равенство

    bool operator==(const EvenNumberIterator& other) const {

        return current == other.current;

    }

    // Операция сравнения на неравенство

    bool operator!=(const EvenNumberIterator& other) const {

        return current != other.current;

    }

    // Операция префиксного инкремента

    EvenNumberIterator& operator++() {

        current += 2;

        return \*this;

    }

    // Операция постфиксного инкремента

    EvenNumberIterator operator++(int) {

        EvenNumberIterator tmp(\*this);

        ++(\*this);

        return tmp;

    }

    // Операция разыменования

    int operator\*() const {

        return current;

    }

};

int sumOfSquaresOfEvenNumbers(int N) {

    int sum = 0;

    // Инициализация итератора для четных чисел от 2 до N

    EvenNumberIterator begin(2, N);

    EvenNumberIterator end(N + 2, N);

    // Перебор четных чисел с помощью итератора

    for (EvenNumberIterator it = begin; it != end; ++it) {

        sum += \*it \* \*it;

    }

    return sum;

}

int main() {

    int N;

    cout << "Введите целое число N: ";

    cin >> N;

    int result = sumOfSquaresOfEvenNumbers(N);

    cout << "Сумма квадратов четных чисел в диапазоне от 1 до " << N << " равна: " << result << endl;

    return 0;

}