МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

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

Кафедра защиты информации

**

**ОТЧЁТ**

**по лабораторной работе №5**

**«**Standart Template Library**»**

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

Выполнил:Проверил:

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

*Клаус В. А. Исаев Г. А.*

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

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

(подпись) (подпись)

Новосибирск 2024

**Цели и задачи работы:** программирование и отладка программ формирования и обработки контейнеров, комбинации контейнеров.

**Задание к работе:** Написать программу решения задачи в соответствии с индивидуальным вариантом.

**Задания:**

1. Реализовать программу для учета товаров на складе.
2. Реализовать программу для электронной очереди в поликлинике.
3. Необходимо реализовать систему хранения и обработки информации по заданию: **Вариант 3** (График движения самолетов).
4. Реализовать программу для автоматизации работы расписания занятий. **Вариант 3.**

**Задание 1**

**С++:**

#include <iostream>

#include <vector>

#include <map>

#include <iomanip>

#include <windows.h>

#include <algorithm>

using namespace std;

struct item { // продукт

    string name;

    int quantity;

};

class sklad { // склад

private:

    map<string, item> storage; // хранилище

    vector<char> zonees; // список зон

public:

    void add(string address, string name, int quantity) { // ф-ия добавления товара

        if (storage.count(address) != 0) { // проверка есть ли такая ячейка

            if (storage[address].name != name && storage[address].name != "") { // проверка, заняты ли ячейка другим продуктом

                cout << "Ошибка, ячейка занята!" << endl;

            } else {

                if (storage[address].quantity + quantity <= 10) { // проверка, превышает ли кол-во продуктов при добавлении число 10

                    storage[address] = {name, storage[address].quantity + quantity};

                } else {

                    cout << "Ошибка, не хватает места в ячейке!" << endl;

                }

            }

        } else { // если нет ячейки, то ошибка

            cout << "Ошибка, нет такой ячейки!" << endl;

        }

    }

    void remove(string address, int quantity) { // ф-ия удаления товара

        if (storage.count(address) != 0) { // проверка есть ли такая ячейка

            if (storage[address].quantity - quantity == 0) { // проверка, если при удалении продуктов стало 0, то освободим ячейку от продукта

                storage[address] = {"", 0};

            } else if (storage[address].quantity - quantity > 0) { // проверка, если при удалении продуктов их будет > 0, то выполняем

                storage[address].quantity -= quantity;

            } else { // если < 0, то продуктов не хватает

                cout << "Ошибка, не хватает продуктов для удаления!" << endl;

            }

        } else {

            cout << "Ошибка, нет такой ячейки!" << endl;

        }

    }

    void info () { // ф-ия вызова информации

        cout << "Адресс\t" << "Продукт\t" << "Кол-во" << endl; // информация по загруженным ячейкам

        float countzanyato = 0.0;

        float countzonezanyato;

        for (auto i : storage) {

            if (i.second.quantity != 0) { // вывод непустых ячеек

                cout << i.first << "\t" << i.second.name << "\t" << i.second.quantity << endl;

                countzanyato++;

            }

        }

        float countvsego = 0.0; // информация по общей загруженности склада

        cout << "Загруженность склада: ";

        for (auto i : storage) { // считаем всего ячеек

            countvsego++;

        }

        float zagrsklad = (countzanyato/countvsego)\*100; // высчитываем загруженность склада

        cout << zagrsklad << "%" << endl;

        float countvsegozones = countvsego / zonees.size(); // Информация по общей загруженности каждой зоны

        vector<float> countzanyatozones(zonees.size());

        cout << "Загруженность каждой зоны:" << endl;

        for (int i = 0; i < zonees.size(); ++i) {

            for (auto j : storage) {

                if(j.first[0] == zonees[i] && j.second.quantity > 0) { // считаем занятые зоны

                    countzanyatozones[i]++;

                }

            }

        }

        float zagrzones;

        for (int i = 0; i < countzanyatozones.size(); ++i) {

            cout << "Зона " << zonees[i] << ": ";

            zagrzones = (countzanyatozones[i]/countvsegozones) \* 100;

            cout << zagrzones << "%" << endl;

        }

        cout << "Пустые ячейки: "; // информация по пустым ячейкам

        for (auto i : storage) {

            if (i.second.quantity == 0) {

                cout << i.first << " ";

            }

        }

        cout << endl;

    }

    void GenerationAddress() { // формируем адресса склада

        vector<char> zones = {'A', 'B', 'C'};

        zonees = zones;

        for (auto i : zones) {

            for (int polka = 1; polka <= 5; ++polka) { // проходим по каждой полке

                for (int section = 1; section <= 5; ++section) { // проходим по каждой секции

                    for (int stelaj = 1; stelaj <= 3; ++stelaj) { // проходим по каждому стелажу

                        // Создаем адресса

                        stringstream s;

                        s << i << stelaj << section << polka;

                        storage[s.str()] = {"", 0};

                    }

                }

            }

        }

    }

};

int main(){

    SetConsoleCP (1251);

    SetConsoleOutputCP (1251);

    sklad warehouse;

    item tovar;

    string address;

    warehouse.GenerationAddress(); // формируем склад(зоны A/B/C, 3 стелажа, 5 секций, 5 полок)

    cout << "Выберите команду: ADD, REMOVE, INFO, EXIT для выхода: "; // работа с командами

    string command;

    bool k = true;

    while (k == true) {

        cin >> command;

        if (command == "ADD") {

            cin >> address >> tovar.name >> tovar.quantity;

            warehouse.add(address, tovar.name, tovar.quantity);

        } else if (command == "REMOVE") {

            cin >> address >> tovar.quantity;

            warehouse.remove(address, tovar.quantity);

        } else if (command == "EXIT") {

            cout << "Выход из программы..." << endl;

            k = false;

        } else if (command == "INFO") {

            warehouse.info();

        } else {

            cout << "Ошибка команды" << endl;

        }

    }

    return 0;

}

**Input:**

**Выберите команду: ADD, REMOVE, INFO, EXIT для выхода: ADD A111 apple 10**

**ADD A112 orange 10**

**REMOVE A112 5**

**INFO**

**Адресс Продукт Кол-во**

**A111 apple 10**

**A112 orange 5**

**Загруженность склада: 0.888889%**

**Загруженность каждой зоны:**

**Зона A: 2.66667%**

**Зона B: 0%**

**Зона C: 0%**

**Пустые ячейки: A113 A114 A115 A121 A122 A123 A124 A125 A131 A132 A133 A134 A135 A141 A142 A143 A144 A145 A151 A152 A153 A154 A155 A211 A212 A213 A214 A215 A221 A222 A223 A224 A225 A231 A232 A233 A234 A235 A241 A242 A243 A244 A245 A251 A252 A253 A254 A255 A311 A312 A313 A314 A315 A321 A322 A323 A324 A325 A331 A332 A333 A334 A335 A341 A342 A343 A344 A345 A351 A352 A353 A354 A355 B111 B112 B113 B114 B115 B121 B122 B123 B124 B125 B131 B132 B133 B134 B135 B141 B142 B143 B144 B145 B151 B152 B153 B154 B155 B211 B212 B213 B214 B215 B221 B222 B223 B224 B225 B231 B232 B233 B234 B235 B241 B242 B243 B244 B245 B251 B252 B253 B254 B255 B311 B312 B313 B314 B315 B321 B322 B323 B324 B325 B331 B332 B333 B334 B335 B341 B342 B343 B344 B345 B351 B352 B353 B354 B355 C111 C112 C113 C114 C115 C121 C122 C123 C124 C125 C131 C132 C133 C134 C135 C141 C142 C143 C144 C145 C151 C152 C153 C154 C155 C211 C212 C213 C214 C215 C221 C222 C223 C224 C225 C231 C232 C233 C234 C235 C241 C242 C243 C244 C245 C251 C252 C253 C254 C255 C311 C312 C313 C314 C315 C321 C322 C323 C324 C325 C331 C332 C333 C334 C335 C341 C342 C343 C344 C345 C351 C352 C353 C354 C355**

**Python:**

class Item:

    def \_\_init\_\_(self, name, quantity):

        self.name = name

        self.quantity = quantity

class Sklad:

    def \_\_init\_\_(self):

        self.storage = {}

        self.zones = []

    def add(self, address, name, quantity):

        if address in self.storage:

            if self.storage[address].name != name and self.storage[address].name != "":

                print("Ошибка, ячейка занята!")

            else:

                if self.storage[address].quantity + quantity <= 10:

                    self.storage[address] = Item(name, self.storage[address].quantity + quantity)

                else:

                    print("Ошибка, не хватает места в ячейке!")

        else:

            print("Ошибка, нет такой ячейки!")

    def remove(self, address, quantity):

        if address in self.storage:

            if self.storage[address].quantity - quantity == 0:

                self.storage[address] = Item("", 0)

            elif self.storage[address].quantity - quantity > 0:

                self.storage[address].quantity -= quantity

            else:

                print("Ошибка, не хватает продуктов для удаления!")

        else:

            print("Ошибка, нет такой ячейки!")

    def info(self):

        print("Адресс\tПродукт\tКол-во")

        count\_occupied = 0.0

        for address, item in self.storage.items():

            if item.quantity != 0:

                print(f"{address}\t{item.name}\t{item.quantity}")

                count\_occupied += 1

        total\_count = len(self.storage)

        print("Загруженность склада: ", end="")

        print(f"{(count\_occupied / total\_count) \* 100}%")

        print("Загруженность каждой зоны:")

        total\_zones = total\_count / len(self.zones)

        count\_occupied\_zones = [0.0] \* len(self.zones)

        for i, zone in enumerate(self.zones):

            for address, item in self.storage.items():

                if address[0] == zone and item.quantity > 0:

                    count\_occupied\_zones[i] += 1

            print(f"Зона {zone}: ", end="")

            print(f"{(count\_occupied\_zones[i] / total\_zones) \* 100}%")

        print("Пустые ячейки: ", end="")

        for address, item in self.storage.items():

            if item.quantity == 0:

                print(address, end=" ")

        print()

    def generate\_address(self):

        zones = ['A', 'B', 'C']

        self.zones = zones

        for zone in zones:

            for shelf in range(1, 6):

                for section in range(1, 6):

                    for rack in range(1, 4):

                        address = f"{zone}{rack}{section}{shelf}"

                        self.storage[address] = Item("", 0)

warehouse = Sklad()

warehouse.generate\_address()

print("Выберите команду: ADD, REMOVE, INFO, EXIT для выхода: ")

k = True

while k:

    command = input()

    if command == "ADD":

        address, name, quantity = input().split()

        quantity = int(quantity)

        warehouse.add(address, name, quantity)

    elif command == "REMOVE":

        address, quantity = input().split()

        quantity = int(quantity)

        warehouse.remove(address, quantity)

    elif command == "EXIT":

        print("Выход из программы...")

        k = False

    elif command == "INFO":

        warehouse.info()

    else:

        print("Ошибка команды")

**Input:**

**Выберите команду: ADD, REMOVE, INFO, EXIT для выхода: ADD A111 apple 10**

**ADD A112 orange 10**

**REMOVE A112 5**

**INFO**

**Адресс Продукт Кол-во**

**A111 apple 10**

**A112 orange 5**

**Загруженность склада: 0.888889%**

**Загруженность каждой зоны:**

**Зона A: 2.66667%**

**Зона B: 0%**

**Зона C: 0%**

**Пустые ячейки: A113 A114 A115 A121 A122 A123 A124 A125 A131 A132 A133 A134 A135 A141 A142 A143 A144 A145 A151 A152 A153 A154 A155 A211 A212 A213 A214 A215 A221 A222 A223 A224 A225 A231 A232 A233 A234 A235 A241 A242 A243 A244 A245 A251 A252 A253 A254 A255 A311 A312 A313 A314 A315 A321 A322 A323 A324 A325 A331 A332 A333 A334 A335 A341 A342 A343 A344 A345 A351 A352 A353 A354 A355 B111 B112 B113 B114 B115 B121 B122 B123 B124 B125 B131 B132 B133 B134 B135 B141 B142 B143 B144 B145 B151 B152 B153 B154 B155 B211 B212 B213 B214 B215 B221 B222 B223 B224 B225 B231 B232 B233 B234 B235 B241 B242 B243 B244 B245 B251 B252 B253 B254 B255 B311 B312 B313 B314 B315 B321 B322 B323 B324 B325 B331 B332 B333 B334 B335 B341 B342 B343 B344 B345 B351 B352 B353 B354 B355 C111 C112 C113 C114 C115 C121 C122 C123 C124 C125 C131 C132 C133 C134 C135 C141 C142 C143 C144 C145 C151 C152 C153 C154 C155 C211 C212 C213 C214 C215 C221 C222 C223 C224 C225 C231 C232 C233 C234 C235 C241 C242 C243 C244 C245 C251 C252 C253 C254 C255 C311 C312 C313 C314 C315 C321 C322 C323 C324 C325 C331 C332 C333 C334 C335 C341 C342 C343 C344 C345 C351 C352 C353 C354 C355**

**Задание 2**

**С++:**

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

void queuePeoples(vector<pair<int, int>>& visitors, int numWindows) {

    vector<int> windowTimes(numWindows, 0);

    vector<int> windowNumbers(numWindows);

    vector<vector<int>> WINDOWS(numWindows);

    // Работа с номерами окон

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

        windowNumbers[i] = i + 1;

    }

    int numNextPeople = 0; // Номер следующего посетителя

    while (!visitors.empty()) {

        // Находим посетителя с самым большим временем посещения

        auto maxPeopleIndex = max\_element(visitors.begin(), visitors.end()) - visitors.begin();

        // Находим индекс окна с минимальным временем работы

        auto minWindowIndex = min\_element(windowTimes.begin(), windowTimes.end()) - windowTimes.begin();

        WINDOWS[minWindowIndex].push\_back(visitors[maxPeopleIndex].second);

        windowTimes[minWindowIndex] += visitors[maxPeopleIndex].first; // Увеличиваем время работы окна на время обслуживания посетителя

        visitors.erase(visitors.begin() + maxPeopleIndex); // Удаляем посетителя из очереди

        numNextPeople++;

    }

    for (int i = 0; i < WINDOWS.size(); ++i) {

        cout << "Окно " << i+1 << "(" << windowTimes[i] << " минут): ";

        for (auto elem : WINDOWS[i]) {

            cout << "T" << elem << " ";

        }

        cout << endl;

    }

}

int main(){

    setlocale(LC\_ALL, "Rus");

    vector<pair<int, int>> Peoples; // вектор пара, в котором будут хранится время и талоны посетителей

    cout << ">>> Введите кол-во рабочих окон:" << endl;

    cout << "<<< ";

    int NumsWindows;

    cin >> NumsWindows;

    cout << ">>> Для получения талона введите: ENQUEUE и время посещения\n\tДля формирования очереди введите: DISTRIBUTE\n\t\tДля выхода введите: EXIT" << endl;

    string command;

    bool k = true; // условие выхода из while

    int time; // время посетителя

    int talon = 100; // талон посетителя

    while (k == true) {

        cout << "<<< ";

        cin >> command;

        if (command == "ENQUEUE") {

            cin >> time;

            talon += 10;

            cout << ">>> T" << talon << endl;

            Peoples.push\_back({time, talon});

        } else if (command == "DISTRIBUTE") {

            queuePeoples(Peoples, NumsWindows); // ф-ия формирования очереди

        } else if (command == "EXIT") {

            cout << ">>> Выход..." << endl;

            k = false;

        } else {

            cout << ">>> Команда введена неверно" << endl;

        }

    }

    return 0;

}

**INPUT:**

**>>> Введите кол-во рабочих окон:**

**<<< 2**

**>>> Для получения талона введите: ENQUEUE и время посещения**

**Для формирования очереди введите: DISTRIBUTE**

**Для выхода введите: EXIT**

**<<< ENQUEUE 1**

**>>> T110**

**<<< ENQUEUE 1**

**>>> T120**

**<<< ENQUEUE 1**

**>>> T130**

**<<< ENQUEUE 10**

**>>> T140**

**<<< DISTRIBUTE**

**Окно 1(10 минут): T140**

**Окно 2(3 минут): T130 T120 T110**

**Python:**

from heapq import heappop, heappush

def queue\_peoples(visitors, num\_windows):

    window\_times = [0] \* num\_windows

    windows = [[] for \_ in range(num\_windows)]

    visitors = [(-time, -talon) for time, talon in visitors]

    heapq.heapify(visitors)

    while visitors:

        time, talon = heappop(visitors)

        min\_window\_index = window\_times.index(min(window\_times))

        windows[min\_window\_index].append(-talon)

        window\_times[min\_window\_index] -= time

    for i, window in enumerate(windows):

        print(f"Окно {i+1} ({window\_times[i]} минут):", ' '.join(f"T{-talon}" for talon in window))

peoples = []

print(">>> Введите кол-во рабочих окон:")

num\_windows = int(input("<<< "))

print(">>> Для получения талона введите: ENQUEUE и время посещения\n\tДля формирования очереди введите: DISTRIBUTE\n\t\tДля выхода введите: EXIT")

talon = 100

while True:

    command = input("<<< ")

    if command == "ENQUEUE":

        time = int(input())

        talon += 10

        print(f">>> T{talon}")

        heappush(peoples, (-time, -talon))

    elif command == "DISTRIBUTE":

        queue\_peoples(peoples, num\_windows)

    elif command == "EXIT":

        print(">>> Выход...")

        break

    else:

        print(">>> Команда введена неверно")

**INPUT:**

**>>> Введите кол-во рабочих окон:**

**<<< 2**

**>>> Для получения талона введите: ENQUEUE и время посещения**

**Для формирования очереди введите: DISTRIBUTE**

**Для выхода введите: EXIT**

**<<< ENQUEUE 1**

**>>> T110**

**<<< ENQUEUE 1**

**>>> T120**

**<<< ENQUEUE 1**

**>>> T130**

**<<< ENQUEUE 10**

**>>> T140**

**<<< DISTRIBUTE**

**Окно 1(10 минут): T140**

**Окно 2(3 минут): T130 T120 T110**

**Задание 3:**

**C++:**

**// plane\_system.h**

#pragma once

#include <iostream>

#include <vector>

#include <string>

#include <map>

#include <algorithm>

using namespace std;

void create\_plane(map<string, vector<string>>& kal);

void planes\_for\_town(const map<string, vector<string>>& kal, const string& town);

void towns\_for\_plane(const map<string, vector<string>>& kal, string plane);

void planes(const map<string, vector<string>>& kal);

enum class Type {

    CREATE\_PLANE,

    PLANES\_FOR\_TOWN,

    TOWNS\_FOR\_PLANE,

    PLANES,

    EXIT,

    OTHERS

};

**// plane\_system.cpp**

#include "plane\_system.h"

void create\_plane(map<string, vector<string>>& kal) { // создание самолета

    cout << "Введите название самолета: ";

    int numtown;

    string plane;

    while (true) {

        cin >> plane;

        if (kal.find(plane) != kal.end()) {

            cout << "Номер самолета " << plane << " уже занят, выберите другой.";

        } else break;

    }

    cout << "Введите число городов: ";

    while (true) {

        cin >> numtown;

        if (numtown < 2) {

            cout << "Ошибка! Минимум 2 города для ввода." << endl;

            cin.clear();

            cin.get();

        } else if (cin.fail()) {

            cout << "Ошибка. Нельзя использовать буквы." << endl;

            cin.clear();

            cin.get();

        } else break;

    }

    // ввод городов

    string town;

    vector<string> numplanetown;

    bool check = false;

    int iter = 0;

    while (true) {

        if (iter == numtown) {

            break;

        }

        cout << "Введите город: ";

        cin >> town;

        check = false;

        for (auto elem : numplanetown) {

            if (elem == town) {

                check = true;

                break;

            }

        }

        if (check) {

            cout << "Ошибка, город повторяется!" << endl;

        } else {

            numplanetown.push\_back(town);

            iter++;

        }

    }

    kal[plane] = numplanetown;

}

void planes\_for\_town(const map<string, vector<string>>& kal, const string& town) { // вывод самолетов, пролетающих город

    cout << "Самолеты летающие через " << town << ":" << endl;

    bool found = false;

    for (auto pair : kal) {

        const vector<string>& towns = pair.second;

        for (string t : towns) {

            if (t == town) {

                cout << "Самолет " << pair.first << endl;

                found = true;

                break;

            }

        }

    }

    if (!found) {

        cout << "Нет пролетающих самолетов " << town << endl;

    }

}

void towns\_for\_plane(const map<string, vector<string>>& kal, string plane) { // вывод городов, которые пролетает самолет

    if (kal.find(plane) != kal.end()) {

        const vector<string>& towns = kal.at(plane);

        cout << "Города, через которые мы пролетели на самолете " << plane << ":" << endl;

        for (string town : towns) {

            cout << town << endl;

            cout << "Самолеты летающие через " << town << ":" << endl; //

            bool found = false;

            for (auto pair : kal) {

                const vector<string>& towns = pair.second;

                for (string t : towns) {

                    if (t == town) {

                        if (pair.first != plane) {

                            cout << "Самолет " << pair.first << endl;

                            found = true;

                            break;

                        }

                    }

                }

            }

            if (!found) {

                cout << "Нет пролетающих самолетов " << town << endl;

            }

        }

    } else {

        cout << "Самолеты " << plane << " не был найден." << endl;

    }

}

void planes(const map<string, vector<string>>& kal) { // вывод информации по самолетам

    for (auto pair : kal) {

        string plane = pair.first;

        cout << "Самолет " << plane << " останавливается в этих городах:" << endl;

        const vector<string>& towns = pair.second;

        for (string town : towns) {

            cout << town << endl;

        }

        cout << endl;

    }

}

**// main.cpp**

#include "plane\_system.h"

#include "plane\_system.cpp"

int main() {

    setlocale(LC\_ALL, "Rus");

    cout << "Все команды: CREATE\_PLANE, PLANES\_FOR\_TOWN, TOWNS\_FOR\_PLANE, PLANES, EXIT" << endl;

    string comand;

    string town;

    string plane;

    map<string, vector<string>> kal;

    Type comands;

    while (true) {

        cout << "Введите команду: ";

        cin >> comand;

        if (comand == "CREATE\_PLANE") comands = Type::CREATE\_PLANE;

        else if (comand == "PLANES\_FOR\_TOWN") comands = Type::PLANES\_FOR\_TOWN;

        else if (comand == "TOWNS\_FOR\_PLANE") comands = Type::TOWNS\_FOR\_PLANE;

        else if (comand == "PLANES") comands = Type::PLANES;

        else if (comand == "EXIT") comands = Type::EXIT;

        else comands = Type::OTHERS;

        if (comands == Type::CREATE\_PLANE) {

            create\_plane(kal);

        } else if (comands == Type::PLANES) {

            planes(kal);

        } else if (comands == Type::PLANES\_FOR\_TOWN) {

            cout << "Введите название города: ";

            cin >> town;

            planes\_for\_town(kal, town);

        } else if (comands == Type::TOWNS\_FOR\_PLANE) {

            cout << "Введите номер самолета: ";

            cin >> plane;

            towns\_for\_plane(kal, plane);

        } else if (comands == Type::EXIT) {

            exit(0);

        } else {

            cout << "Неверный ввод" << endl;

        }

    }

    return 0;

}

**INPUT:**

**Все команды: CREATE\_PLANE, PLANES\_FOR\_TOWN, TOWNS\_FOR\_PLANE, PLANES, EXIT**

**Введите команду: CREATE\_PLANE**

**Введите название самолета: S7**

**Введите число городов: 2**

**Название города: nsk**

**zeya**

**Введите команду: CREATE\_PLANE**

**Введите название самолета: S10**

**Введите число городов: 1**

**Название города: zeya**

**Введите команду: PLANES\_FOR\_TOWN**

**Введите название города: zeya**

**Самолеты летающие через zeya:**

**Самолет S10**

**Самолет S7**

**Введите команду: TOWNS\_FOR\_PLANE**

**Введите номер самолета: S10**

**Города, через которые мы пролетели на самолете S10:**

**zeya**

**Самолеты летающие через zeya:**

**Самолет S7**

**Введите команду: PLANES**

**Самолет S10 останавливается в этих городах:**

**zeya**

**Самолет S7 останавливается в этих городах:**

**nsk**

**zeya**

**Python:**

**def create\_plane(kal):**

**plane = input("Введите название самолета: ")**

**while plane in kal:**

**plane = input("Номер самолета уже занят, выберите другой: ")**

**numtown = input("Введите число городов: ")**

**while not numtown.isdigit():**

**numtown = input("Ошибка. Нельзя использовать буквы. Введите число городов: ")**

**numtown = int(numtown)**

**numplanetown = [input("Название города: ") for \_ in range(numtown)]**

**kal[plane] = numplanetown**

**def planes\_for\_town(kal, town):**

**print(f"Самолеты летающие через {town}:")**

**found = False**

**for plane, towns in kal.items():**

**if town in towns:**

**print(f"Самолет {plane}")**

**found = True**

**if not found:**

**print(f"Нет пролетающих самолетов через {town}")**

**def towns\_for\_plane(kal, plane):**

**if plane in kal:**

**towns = kal[plane]**

**print(f"Города, через которые пролетает самолет {plane}:")**

**for town in towns:**

**print(town)**

**else:**

**print(f"Самолет {plane} не найден.")**

**def planes(kal):**

**for plane, towns in kal.items():**

**print(f"Самолет {plane} останавливается в этих городах:")**

**for town in towns:**

**print(town)**

**print()**

**print("Все команды: CREATE\_PLANE, PLANES\_FOR\_TOWN, TOWNS\_FOR\_PLANE, PLANES, EXIT")**

**kal = {}**

**while True:**

**comands = input("Введите команду: ")**

**if comands == "CREATE\_PLANE":**

**create\_plane(kal)**

**elif comands == "PLANES":**

**planes(kal)**

**elif comands == "PLANES\_FOR\_TOWN":**

**town = input("Введите название города: ")**

**planes\_for\_town(kal, town)**

**elif comands == "TOWNS\_FOR\_PLANE":**

**plane = input("Введите номер самолета: ")**

**towns\_for\_plane(kal, plane)**

**elif comands == "EXIT":**

**break**

**else:**

**print("Неверный ввод")**

**INPUT:**

**Все команды: CREATE\_PLANE, PLANES\_FOR\_TOWN, TOWNS\_FOR\_PLANE, PLANES, EXIT**

**Введите команду: CREATE\_PLANE**

**Введите название самолета: S7**

**Введите число городов: 2**

**Название города: nsk**

**zeya**

**Введите команду: CREATE\_PLANE**

**Введите название самолета: S10**

**Введите число городов: 1**

**Название города: zeya**

**Введите команду: PLANES\_FOR\_TOWN**

**Введите название города: zeya**

**Самолеты летающие через zeya:**

**Самолет S10**

**Самолет S7**

**Введите команду: TOWNS\_FOR\_PLANE**

**Введите номер самолета: S10**

**Города, через которые мы пролетели на самолете S10:**

**zeya**

**Самолеты летающие через zeya:**

**Самолет S7**

**Введите команду: PLANES**

**Самолет S10 останавливается в этих городах:**

**zeya**

**Самолет S7 останавливается в этих городах:**

**nsk**

**zeya**

**Задание 4**

**C++:**

#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

using namespace std;

int main() {

    setlocale(LC\_ALL, "Russian");

    vector<pair<int, string>> schedule;

    while (true) {

        int N;

        cout << "Введите число операций: ";

        cin >> N;

        string operation;

        for (int i = 0; i < N; ++i) {

            cout << "Введите операцию: ";

            cin >> operation;

            if (operation == "CLASS") {

                string discipline;

                int day;

                cout << "Введите номер дня и название предмета: ";

                cin >> day >> discipline;

                schedule.push\_back(make\_pair(day, discipline)); // Добавляем занятие в расписание

            } else if (operation == "NEXT") {

                int lastDay = 29; // Предпоследний день текущего месяца

                for (auto& cls : schedule) {

                    cls.first = lastDay; // Переносим все занятия на предпоследний день текущего месяца

                }

                cout << "Перенос всех занятий на предпоследний день текущего месяца выполнен." << endl;

            } else if (operation == "VIEW") {

                int day;

                cout << "Введите номер дня: ";

                cin >> day;

                // Используем алгоритм find\_if для поиска первого занятия на указанный день

                auto it = find\_if(schedule.begin(), schedule.end(), [day](const pair<int, string>& cls) {

                    return cls.first == day;

                });

                cout << "В " << day << " день ";

                if (it != schedule.end()) { // Если занятия нашлись на указанный день

                    cout << "1 занятие в университете: " << it->second;

                    ++it;

                    while (it != schedule.end() && it->first == day) { // Выводим все занятия на этот день

                        cout << ", " << it->second;

                        ++it;

                    }

                } else {

                    cout << "У нас свободный день!"; // Если занятий нет на указанный день

                }

                cout << endl;

            } else if (operation == "EXIT") {

                cout << "Программа завершает работу." << endl;

                return 0;

            }

        }

    }

    return 0;

}

**INPUT:**

**Введите число N: 2**

**Введите операцию: CLASS**

**Введите номер дня и название предмета: 10 math**

**Введите операцию: VIEW**

**Введите номер дня: 10**

**В 10 день 1 занятие в университете: math**

**Python:**

schedule = []

while True:

    N = int(input("Введите число операций: "))

    for \_ in range(N):

        operation = input("Введите операцию: ")

        if operation == "CLASS":

            day, discipline = input("Введите номер дня и название предмета: ").split()

            day = int(day)

            schedule.append((day, discipline))  # Добавляем занятие в расписание

        elif operation == "NEXT":

            last\_day = 29  # Предпоследний день текущего месяца

            schedule = [(last\_day, cls[1]) for cls in schedule]  # Переносим все занятия на предпоследний день

            print("Перенос всех занятий на предпоследний день текущего месяца выполнен.")

        elif operation == "VIEW":

            day = int(input("Введите номер дня: "))

            classes\_on\_day = [cls for cls in schedule if cls[0] == day]

            print(f"В {day} день", end=" ")

            if classes\_on\_day:

                print(f"1 занятие в университете: {classes\_on\_day[0][1]}", end="")

                for cls in classes\_on\_day[1:]:

                    print(f", {cls[1]}", end="")

            else:

                print("У нас свободный день!", end="")

            print()

        elif operation == "EXIT":

            print("Программа завершает работу.")

**INPUT:**

**Введите число N: 2**

**Введите операцию: CLASS**

**Введите номер дня и название предмета: 10 math**

**Введите операцию: VIEW**

**Введите номер дня: 10**

**В 10 день 1 занятие в университете: math**

**Вывод:**

В ходе выполнения лабораторной работы №5 я научился реализовывать различные базы данных и автоматизировать их. Научился работать с многофайловым проектом, что в дальнейшем мне очень сильно пригодится.

**Ссылка на репозиторий:** [нажмите](https://github.com/JamesSevil/laba5)