МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

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

Кафедра защиты информации

**

**ЛАБОРАТОРНАЯ РАБОТА №5**

**по дисциплине: «Программирование»**

**на тему: «**Standard template library**»**

|  |  |
| --- | --- |
| Выполнил:  Студент гр. «АБ-420», «АВТФ»,  *Никифоров Максим Владимирович*  «\_15\_» апреля 2025г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | Проверил:  *Ассистент кафедры ЗИ*  *Исаев Глеб Андреевичн*  «\_\_\_» \_\_\_\_\_\_ 2025г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Новосибирск 2025

СОДЕРЖАНИЕ

Оглавление

[Ход выполнения 3](#_Toc195658761)

[Задание 1. 3](#_Toc195658762)

[Реализация на C++. 3](#_Toc195658763)

[Реализация на Go. 6](#_Toc195658764)

[Задание 2. 11](#_Toc195658765)

[Реализация на Go. 12](#_Toc195658766)

[Задание 3. 14](#_Toc195658767)

[Реализация на C++. 14](#_Toc195658768)

[Задание 4. 19](#_Toc195658769)

[Реализация на C++. 19](#_Toc195658770)

[Вывод 22](#_Toc195658771)

# Ход выполнения

## Задание 1.

### Реализация на C++.

#include <iostream>

#include <map>

#include <vector>

using namespace std;

struct ticketS {

string id;

int duration;

};

string pushTicket(vector<ticketS>& tickets, const int time) {

int ticketN;

if (!tickets.empty()) {

ticketN = stoi(tickets.back().id.substr(1, string::npos));

} else {

ticketN = 0;

}

string ticketId = "T" + to\_string(ticketN+1);

ticketS newTicket = {ticketId, time};

tickets.push\_back(newTicket);

return newTicket.id;

}

vector<string> splitString(const string& s) {

vector<string> res;

size\_t pos = s.find(' ');

if (pos != string::npos) {

res.push\_back(s.substr(0, pos));

res.push\_back(s.substr(pos + 1));

} else {

res.push\_back(s); // если пробела нет, вернём строку как один элемент

}

return res;

}

int main() {

int threadsAmount;

cout << ">>> Введите количество окон:\n<<< ";

cin >> threadsAmount;

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

vector<ticketS> tickets;

string command;

while(cout << "<<< ", getline(cin,command)) {

vector<string> splitted = splitString(command);

string task = splitted[0];

if (task == "ENQUEUE") {

string ticketId = pushTicket(tickets, stoi(splitted[1]));

cout << ">>> " << ticketId << "\n";

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

vector<pair<vector<ticketS>, int>> queue(threadsAmount);

vector<ticketS> sortedTickets = tickets;

sort(sortedTickets.begin(), sortedTickets.end(), [](const ticketS& a, const ticketS& b) {

return a.duration > b.duration;

});

for (const auto& ticket : sortedTickets) {

auto minIt = queue.begin();

for (auto it = queue.begin(); it != queue.end(); ++it) {

if (it->second < minIt->second) {

minIt = it;

}

}

minIt->first.push\_back(ticket);

minIt->second += ticket.duration;

}

int windN = 0;

for (auto& window : queue) {

cout << ">>> Окно " << windN << " (" << window.second << " минут): ";

windN+=1;

for (auto& ticket : window.first) {

cout << ticket.id << " ";

}

cout << "\n";

}

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

cout << ">>> Executing exit" << endl;

break;

} else {

cout << ">>> Unknown command. Try again\n";

}

}

return 0;

}

Тестирование работы программы. Рисунок 1.

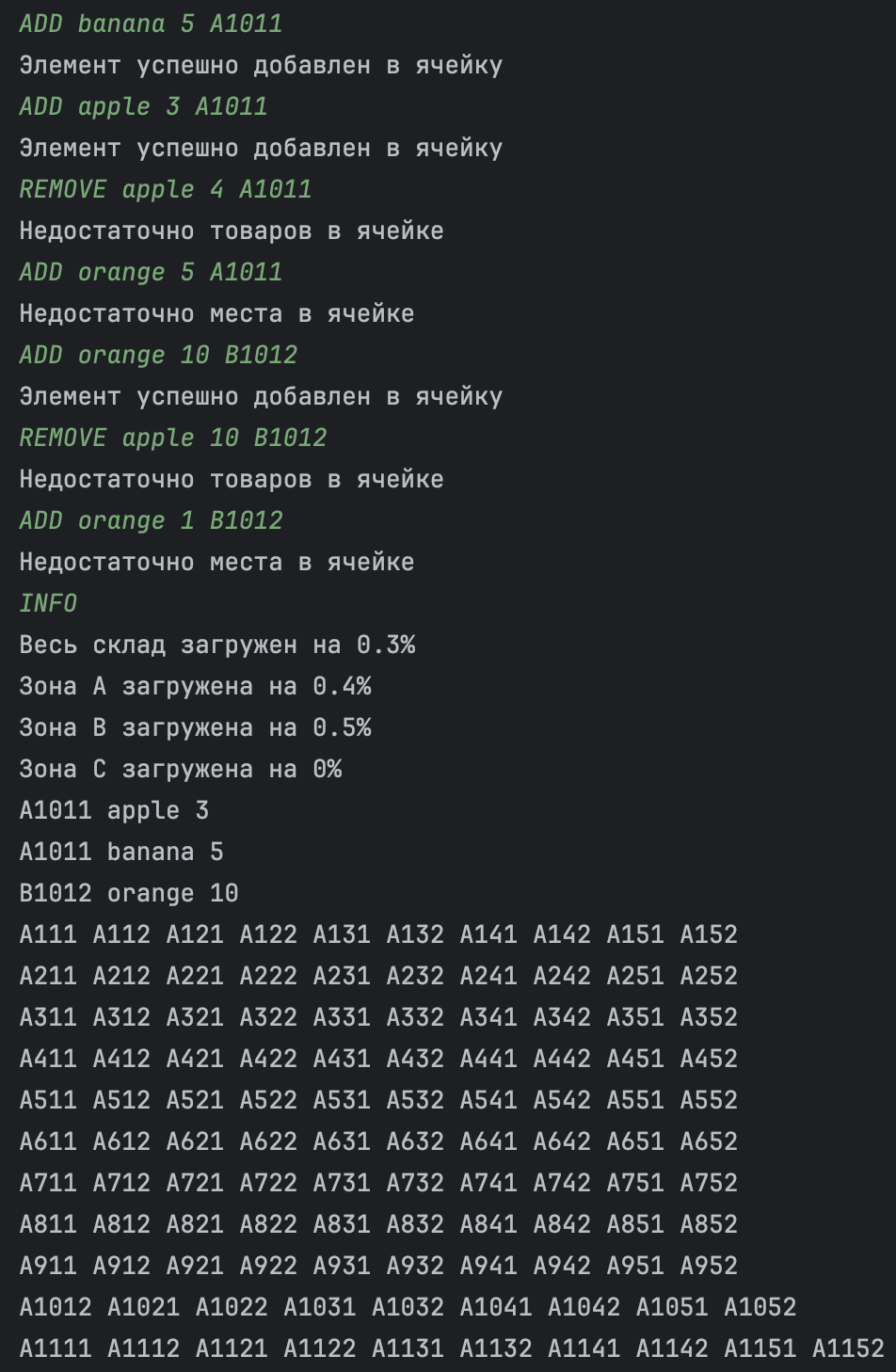


Рисунок 1 ­– Тест программы разными командами.

### Реализация на Go.

package main

import (

"bufio"

"fmt"

"os"

"strconv"

"strings"

)

type Cell struct {

items map[string]int

}

type Storage struct {

zones []string

racks []int

vSects []int

hSects []int

loadedZones map[string]int

totalCapacity int

freeCapacity int

cells map[string]Cell

}

type result int

const (

SUCCESS result = iota

NOCELL

TOOMANY

)

// добавляем товар в ячейку

func add(procStorage \*Storage, item string, amount int, cell string) result {

storageCell, ok := procStorage.cells[cell]

if !ok {

return NOCELL

}

currQty := 0

for \_, qty := range storageCell.items {

currQty += qty

}

if currQty+amount <= 10 {

storageCell.items[item] += amount

procStorage.cells[cell] = storageCell

procStorage.freeCapacity -= amount

zone := string(cell[0])

procStorage.loadedZones[zone] += amount

return SUCCESS

}

return TOOMANY

}

// убираем товар из ячейки

func remove(procStorage \*Storage, item string, amount int, cell string) result {

storageCell, ok := procStorage.cells[cell]

if !ok {

return NOCELL

}

currQty := storageCell.items[item]

if currQty-amount >= 0 {

storageCell.items[item] -= amount

procStorage.cells[cell] = storageCell

procStorage.freeCapacity += amount

zone := string(cell[0])

procStorage.loadedZones[zone] -= amount

return SUCCESS

}

return TOOMANY

}

// выводим данные по складу

func printInfo(procStorage \*Storage) {

totalLoad := float64((procStorage.totalCapacity-procStorage.freeCapacity)\*100) / float64(procStorage.totalCapacity)

fmt.Printf("Весь склад загружен на %.2f%%\n", totalLoad)

zoneAm := len(procStorage.loadedZones)

for zone, load := range procStorage.loadedZones {

zoneLoad := float64(load) / (float64(procStorage.totalCapacity) / float64(zoneAm)) \* 100

fmt.Printf("Зона %s загружена на %.2f%%\n", zone, zoneLoad)

}

for cell, c := range procStorage.cells {

for item, qty := range c.items {

if qty > 0 {

fmt.Println(cell, item, qty)

}

}

}

// пустые ячейки

for \_, zone := range procStorage.zones {

for \_, rack := range procStorage.racks {

for \_, v := range procStorage.vSects {

for \_, h := range procStorage.hSects {

address := fmt.Sprintf("%s%d%d%d", zone, rack, v, h)

if len(procStorage.cells[address].items) == 0 {

fmt.Print(address + " ")

}

}

}

fmt.Println()

}

}

}

// разбиваем строку на части

func splitString(s string, divider string) []string {

return strings.Split(s, divider)

}

func main() {

zones := []string{"A", "B", "C"}

racks := []int{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20}

vSects := []int{1, 2, 3, 4, 5}

hSects := []int{1, 2}

loadedZones := map[string]int{"A": 0, "B": 0, "C": 0}

mainStorage := Storage{

zones: zones,

racks: racks,

vSects: vSects,

hSects: hSects,

loadedZones: loadedZones,

cells: make(map[string]Cell),

}

mainStorage.totalCapacity = len(zones) \* len(racks) \* len(vSects) \* len(hSects) \* 10

mainStorage.freeCapacity = mainStorage.totalCapacity

// генерим адреса ячеек

for \_, zone := range zones {

for \_, rack := range racks {

for \_, v := range vSects {

for \_, h := range hSects {

address := fmt.Sprintf("%s%d%d%d", zone, rack, v, h)

mainStorage.cells[address] = Cell{items: make(map[string]int)}

}

}

}

}

scanner := bufio.NewScanner(os.Stdin)

for scanner.Scan() {

command := scanner.Text()

splitted := splitString(command, " ")

if len(splitted) == 0 {

continue

}

task := splitted[0]

switch task {

case "ADD":

if len(splitted) != 4 {

fmt.Println("Invalid arguments amount. Try again")

continue

}

amount, \_ := strconv.Atoi(splitted[2])

status := add(&mainStorage, splitted[1], amount, splitted[3])

if status == SUCCESS {

fmt.Println("Элемент успешно добавлен в ячейку")

} else if status == NOCELL {

fmt.Println("Ячейка не найдена")

} else if status == TOOMANY {

fmt.Println("Недостаточно места в ячейке")

}

case "REMOVE":

if len(splitted) != 4 {

fmt.Println("Invalid arguments amount. Try again")

continue

}

amount, \_ := strconv.Atoi(splitted[2])

status := remove(&mainStorage, splitted[1], amount, splitted[3])

if status == SUCCESS {

fmt.Println("Товар успешно удален из ячейки")

} else if status == NOCELL {

fmt.Println("Ячейка не найдена")

} else if status == TOOMANY {

fmt.Println("Недостаточно товаров в ячейке")

}

case "INFO":

printInfo(&mainStorage)

case "EXIT":

fmt.Println("Executing exit")

return

default:

fmt.Println("Unknown command. Try again")

}

}

}

Тестирование работы программы. Рисунок 2.

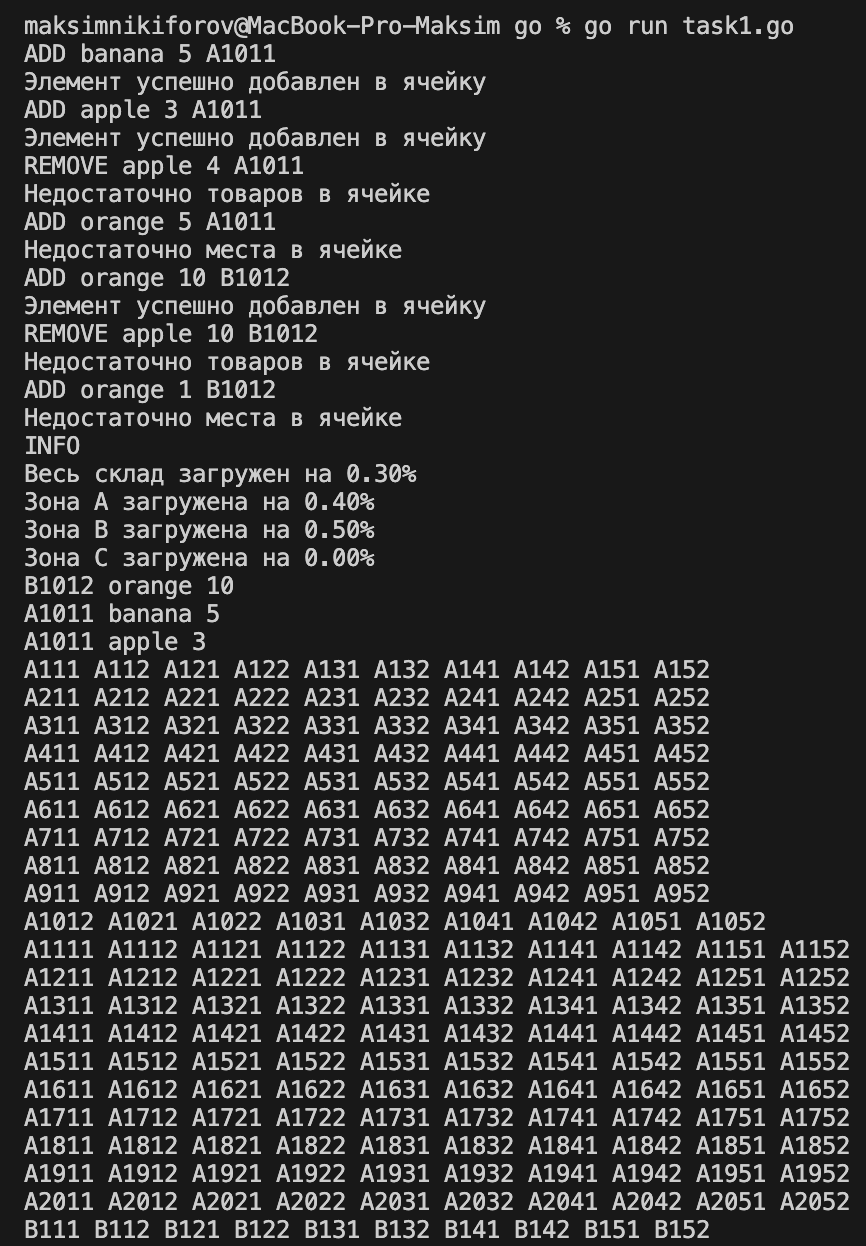


Рисунок 2 ­– Тест программы разными командами.

## Задание 2.

Реализация на C++.

#include <iostream>  
#include <map>  
#include <vector>  
using namespace std;  
  
struct ticketS {  
 string id;  
 int duration;  
};  
string pushTicket(vector<ticketS>& tickets, const int time) {  
 int ticketN;  
 if (!tickets.empty()) {  
 ticketN = stoi(tickets.back().id.substr(1, string::*npos*));  
 } else {  
 ticketN = 0;  
 }  
 string ticketId = "T" + to\_string(ticketN+1);  
 ticketS newTicket = {ticketId, time};  
 tickets.push\_back(newTicket);  
 return newTicket.id;  
}  
  
vector<string> splitString(const string& s) {  
 vector<string> res;  
 size\_t pos = s.find(' ');  
 if (pos != string::*npos*) {  
 res.push\_back(s.substr(0, pos));  
 res.push\_back(s.substr(pos + 1));  
 } else {  
 res.push\_back(s); // если пробела нет, вернём строку как один элемент  
 }  
 return res;  
}  
  
int main() {  
 int threadsAmount;  
 cout << ">>> Введите количество окон:\n<<< ";  
 cin >> threadsAmount;  
  
 cin.clear();  
 cin.ignore(numeric\_limits<streamsize>::*max*(), '\n');  
  
 vector<ticketS> tickets;  
  
 string command;  
 while(cout << "<<< ", getline(cin,command)) {  
 vector<string> splitted = splitString(command);  
 string task = splitted[0];  
  
 if (task == "ENQUEUE") {  
 string ticketId = pushTicket(tickets, stoi(splitted[1]));  
 cout << ">>> " << ticketId << "\n";  
 } else if (task == "DISTRIBUTE") {  
 vector<pair<vector<ticketS>, int>> queue(threadsAmount);  
  
 vector<ticketS> sortedTickets = tickets;  
 sort(sortedTickets.begin(), sortedTickets.end(), [](const ticketS& a, const ticketS& b) {  
 return a.duration > b.duration;  
 });  
  
 for (const auto& ticket : sortedTickets) {  
 auto minIt = queue.begin();  
  
 for (auto it = queue.begin(); it != queue.end(); ++it) {  
 if (it->second < minIt->second) {  
 minIt = it;  
 }  
 }  
  
 minIt->first.push\_back(ticket);  
 minIt->second += ticket.duration;  
 }  
 int windN = 0;  
 for (auto& window : queue) {  
 cout << ">>> Окно " << windN << " (" << window.second << " минут): ";  
 windN+=1;  
 for (auto& ticket : window.first) {  
 cout << ticket.id << " ";  
 }  
 cout << "\n";  
 }  
 } else if (task == "EXIT") {  
 cout << ">>> Executing exit" << endl;  
 break;  
 } else {  
 cout << ">>> Unknown command. Try again\n";  
 }  
 }  
 return 0;  
}

Тестирование работы программы. Рисунок 3.

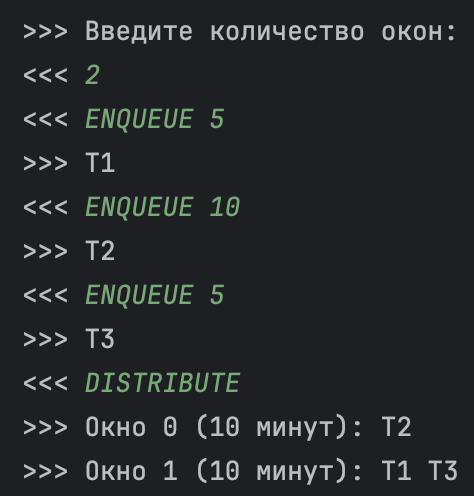


Рисунок 3 – Тест программы по примеру из задания

### Реализация на Go.

package main

import (

"bufio"

"fmt"

"os"

"sort"

"strconv"

"strings"

)

type ticketS struct {

id string

duration int

}

// создаём тикет, добавляем в слайс, возвращаем id

func pushTicket(tickets \*[]ticketS, time int) string {

ticketN := 0

if len(\*tickets) > 0 {

last := (\*tickets)[len(\*tickets)-1]

numPart := last.id[1:]

ticketN, \_ = strconv.Atoi(numPart)

}

ticketId := "T" + strconv.Itoa(ticketN+1)

\*tickets = append(\*tickets, ticketS{ticketId, time})

return ticketId

}

// сплит по пробелу

func splitString(s string) []string {

return strings.SplitN(s, " ", 2)

}

func main() {

scanner := bufio.NewScanner(os.Stdin)

fmt.Print(">>> Введите количество окон:\n<<< ")

scanner.Scan()

threadsAmount, \_ := strconv.Atoi(scanner.Text())

var tickets []ticketS

for {

fmt.Print("<<< ")

scanner.Scan()

command := scanner.Text()

splitted := splitString(command)

task := splitted[0]

if task == "ENQUEUE" {

if len(splitted) != 2 {

fmt.Println(">>> Неверный формат. Используйте: ENQUEUE <время>")

continue

}

time, \_ := strconv.Atoi(splitted[1])

ticketId := pushTicket(&tickets, time)

fmt.Println(">>>", ticketId)

} else if task == "DISTRIBUTE" {

// создаём очередь из пар: [тикеты, сумма времени]

queue := make([][]ticketS, threadsAmount)

totals := make([]int, threadsAmount)

sortedTickets := make([]ticketS, len(tickets))

copy(sortedTickets, tickets)

// сортировка по убыванию времени

sort.Slice(sortedTickets, func(i, j int) bool {

return sortedTickets[i].duration > sortedTickets[j].duration

})

// раскидываем тикеты по окнам с минимальной загруженностью

for \_, ticket := range sortedTickets {

minIdx := 0

for i := 1; i < threadsAmount; i++ {

if totals[i] < totals[minIdx] {

minIdx = i

}

}

queue[minIdx] = append(queue[minIdx], ticket)

totals[minIdx] += ticket.duration

}

for i, window := range queue {

fmt.Printf(">>> Окно %d (%d минут): ", i, totals[i])

for \_, ticket := range window {

fmt.Print(ticket.id, " ")

}

fmt.Println()

}

} else if task == "EXIT" {

fmt.Println(">>> Executing exit")

break

} else {

fmt.Println(">>> Unknown command. Try again")

}

}

}

Тестирование работы программы. Рисунок 4.

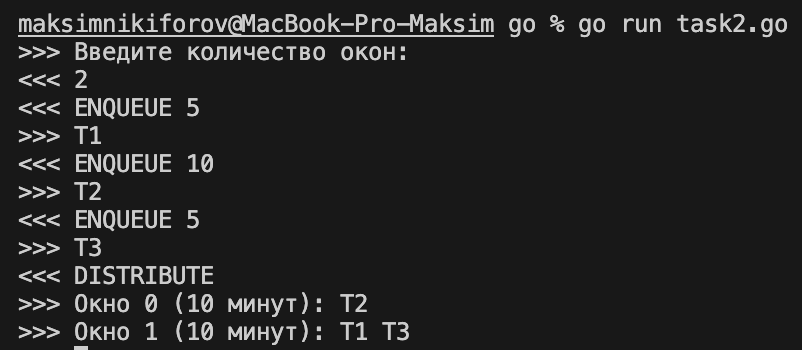


Рисунок 4 – Тест программы по примеру из задания

## Задание 3.

Реализация на C++. Многофайловый проект. Структура:

1. Main.cpp – точка входа, ввод/вывод данных пользователя.
2. Commands.h – заголовки функций, обрабатывающих запросы
3. Commands.cpp – функции, работающие с запросами пользователя.

Файл main.cpp. точка входа, ввод/вывод данных пользователя.

#include <iostream>

#include <vector>

#include "commands.h"

using namespace std;

vector<string> splitString(string s, const string& divider){

vector<string> res;

size\_t pos = 0;

while ((pos = s.find(divider)) != string::npos) {

res.push\_back(s.substr(0, pos));

s.erase(0, pos + divider.length());

}

res.push\_back(s); // остаток строки

return res;

}

Type getCommand(const string& text) {

const map<string, Type> commands = {

{"CREATE\_TRAIN", Type::CREATE\_TRAIN},

{"TRAINS\_FOR\_TOWN", Type::TRAINS\_FOR\_TOWN},

{"TOWNS\_FOR\_TRAIN", Type::TOWNS\_FOR\_TRAIN},

{"TRAINS", Type::TRAINS},

};

return commands.find(text)->second;

}

int main() {

unordered\_map<string, vector<string>> towns;

unordered\_map<string, vector<string>> trains;

string command;

while (cout << "<<< ", getline(cin, command)) {

vector<string> splitted = splitString(command, " ");

Type task = getCommand(splitted[0]);

if (task == Type::CREATE\_TRAIN) {

string trainName = splitted[1];

vector<string> trainTowns;

trainTowns.insert(trainTowns.end(), splitted.begin()+2, splitted.end());

if (trainTowns.size() > 1) {

createTrain(towns, trains, trainName, trainTowns);

} else {

cout << ">>> Too little train stops!\n";

}

} else if (task == Type::TRAINS\_FOR\_TOWN) {

string townName = splitted[1];

vector<string> trForTown = getTrainsForTown(towns,trains,townName);

if (trForTown[0] == "notown") { cout << ">>> Нет такого города!\n"; }

else if (trForTown[0] == "notrains") { cout << ">>> В этом городе поезда не останавливаются!\n"; }

else {

cout << ">>> ";

for (string& train : trForTown) { cout << train << " "; }

cout << endl;

}

} else if (task == Type::TOWNS\_FOR\_TRAIN) {

string train = splitted[1];

unordered\_map<string, vector<string>> townsForTrain = getTownsForTrain(towns,trains, train);

for (auto& [town, trains] : townsForTrain) {

cout << town << ": ";

for (auto& train : trains) {

cout << train << " ";

}

cout << endl;

}

} else if (task == Type::TRAINS) {

printTrainsList(trains);

}

else {

cout << ">>> Unknown command. Try again\n";

}

}

return 0;

}

Файл commands.h – заголовки функций, обрабатывающих запросы.

#include <map>

#include <vector>

#include <string>

using namespace std;

#ifndef COMMANDS\_H

#define COMMANDS\_H

enum class Type {

CREATE\_TRAIN,

TRAINS\_FOR\_TOWN,

TOWNS\_FOR\_TRAIN,

TRAINS

};

void createTrain(unordered\_map<string, vector<string>>& towns, unordered\_map<string, vector<string>>& trains, string trainName, vector<string> trainTowns);

vector<string> getTrainsForTown(unordered\_map<string, vector<string>>& towns, unordered\_map<string, vector<string>>& trains, string townName);

unordered\_map<string, vector<string>> getTownsForTrain(unordered\_map<string, vector<string>>& towns, unordered\_map<string, vector<string>>& trains, string trainName);

void printTrainsList(unordered\_map<string, vector<string>>& trains);

#endif //COMMANDS\_H

Файл Commands.cpp – функции, работающие с запросами пользователя.

#include "commands.h"

#include <vector>

#include <map>

#include <iostream>

using namespace std;

void createTrain(unordered\_map<string, vector<string>>& towns, unordered\_map<string, vector<string>>& trains, string trainName, vector<string> trainTowns) {

trains[trainName] = trainTowns;

for (auto& town : trainTowns) {

towns[town].push\_back(trainName);

}

}

vector<string> getTrainsForTown(unordered\_map<string, vector<string>>& towns, unordered\_map<string, vector<string>>& trains, string townName) {

if (towns.find(townName) == towns.end()) { return {"notown"}; }

return towns[townName];

}

unordered\_map<string, vector<string>> getTownsForTrain(unordered\_map<string, vector<string>>& towns, unordered\_map<string, vector<string>>& trains, string trainName) {

unordered\_map<string, vector<string>> townsForTrain;

if (trains.find(trainName) == trains.end()) { return townsForTrain; }

for (auto& town : trains[trainName]) {

for (auto& train : towns[town]) {

if (train != trainName) {

townsForTrain[town].push\_back(train);

}

}

}

return townsForTrain;

}

void printTrainsList(unordered\_map<string, vector<string>>& trains) {

for (auto& [train, towns] : trains) {

cout << ">>> " << train << ": ";

for (auto& town : towns) {

cout << town << " ";

}

cout << endl;

}

}

Тестирование работы программы. Рисунок 5.

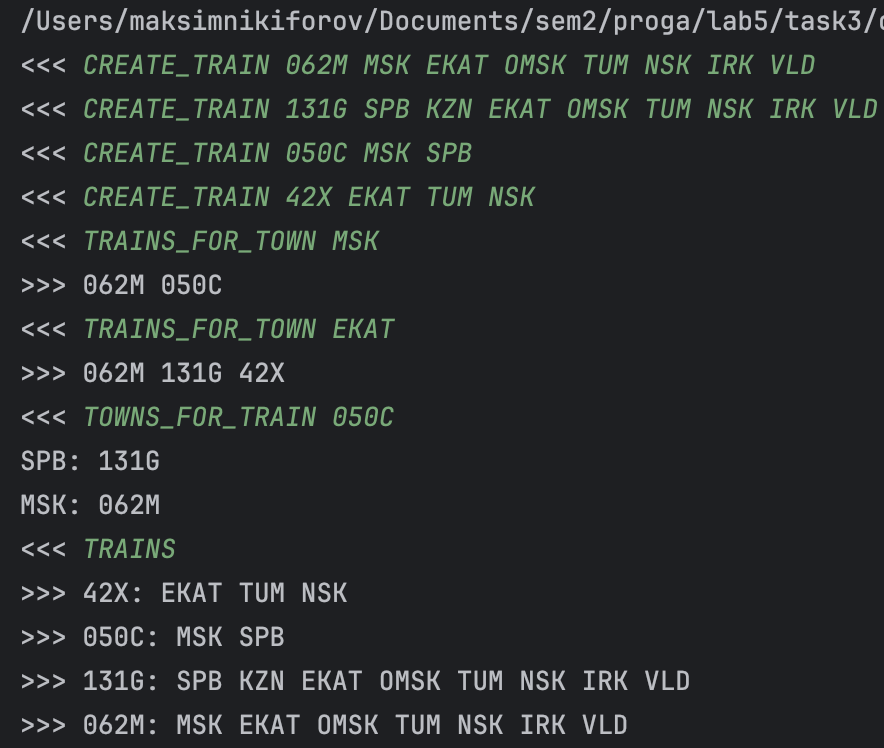


Рисунок 5 – Тест программы с некоторыми реальными поездами.

## Задание 4.

### Реализация на C++.

#include <iostream>  
#include <set>  
using namespace std;  
  
void addFriend(unordered\_map<string, set<string>>& guys, const string& person1, const string& person2) {  
 guys[person1].insert(person2);  
 guys[person2].insert(person1);  
}  
  
int friendsCount (unordered\_map<string, set<string>>& guys, const string& person1) {  
 return guys[person1].size();  
}  
  
bool isFriends (unordered\_map<string, set<string>>& guys, const string& person1, const string& person2) {  
 return find(guys[person1].begin(), guys[person1].end(), person2) != guys[person1].end();  
}  
  
vector<string> splitString(string s, const string& divider){  
 vector<string> res;  
 size\_t pos = 0;  
 while ((pos = s.find(divider)) != string::*npos*) {  
 res.push\_back(s.substr(0, pos));  
 s.erase(0, pos + divider.length());  
 }  
 res.push\_back(s); // остаток строки  
 return res;  
}  
  
int main() {  
 unordered\_map<string, set<string>> people;  
  
 string command;  
 while (cout << "<<< ", getline(cin, command)) {  
 vector<string> splitted = splitString(command, " ");  
 string task = splitted[0];  
  
 if (task == "FRIENDS") {  
 addFriend(people, splitted[1], splitted[2]);  
 cout << ">>> OK\n";  
 } else if (task == "COUNT") {  
 cout << ">>> " << friendsCount(people, splitted[1]) << "\n";  
 } else if (task == "QUESTION") {  
 cout << ">>> ";  
 isFriends(people,splitted[1],splitted[2]) ? cout << "YES" : cout << "NO";  
 cout << endl;  
 } else {  
 cout << ">>> Unknown command. Try again\n";  
 }  
 }  
 return 0;  
}

Тестирование работы программы. Рисунки 6-10.

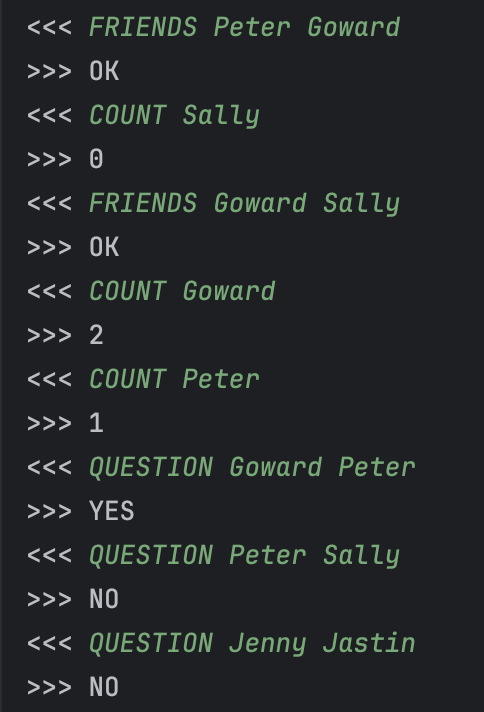


Рисунок 6 – Тест на данных из задания



Рисунок 7 – Тест на придуманных данных



Рисунок 8 – Тест на других придуманных данных

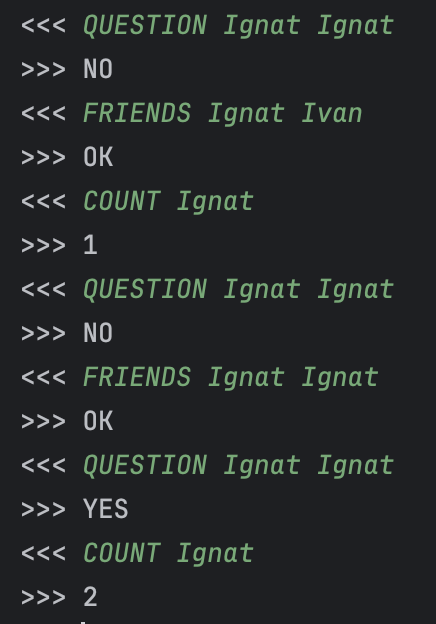


Рисунок 9 – Тест на выдуманных данных



Рисунок 10 – Тест на выдуманных данных.

# Вывод

В ходе данной лабораторной работы была достигнута цель по программированию и отладке программ формирования и обработки контейнеров данных, была изучена работа с комбинациями контейнеров.

Github: ….