Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Электротехнический факультет  
Кафедра «Информационные технологии и автоматизированные системы» направление подготовки: 09.03.04 – «Программная инженерия»

# Лабораторная работа "Хэш-таблица"

Выполнил студент гр. РИС-24-3б

Жиряков Леонид Антонович

Проверил:

Доцент кафедры ИТАС   
Ольга Андреевна Полякова

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (оценка) (подпись)

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

(дата)

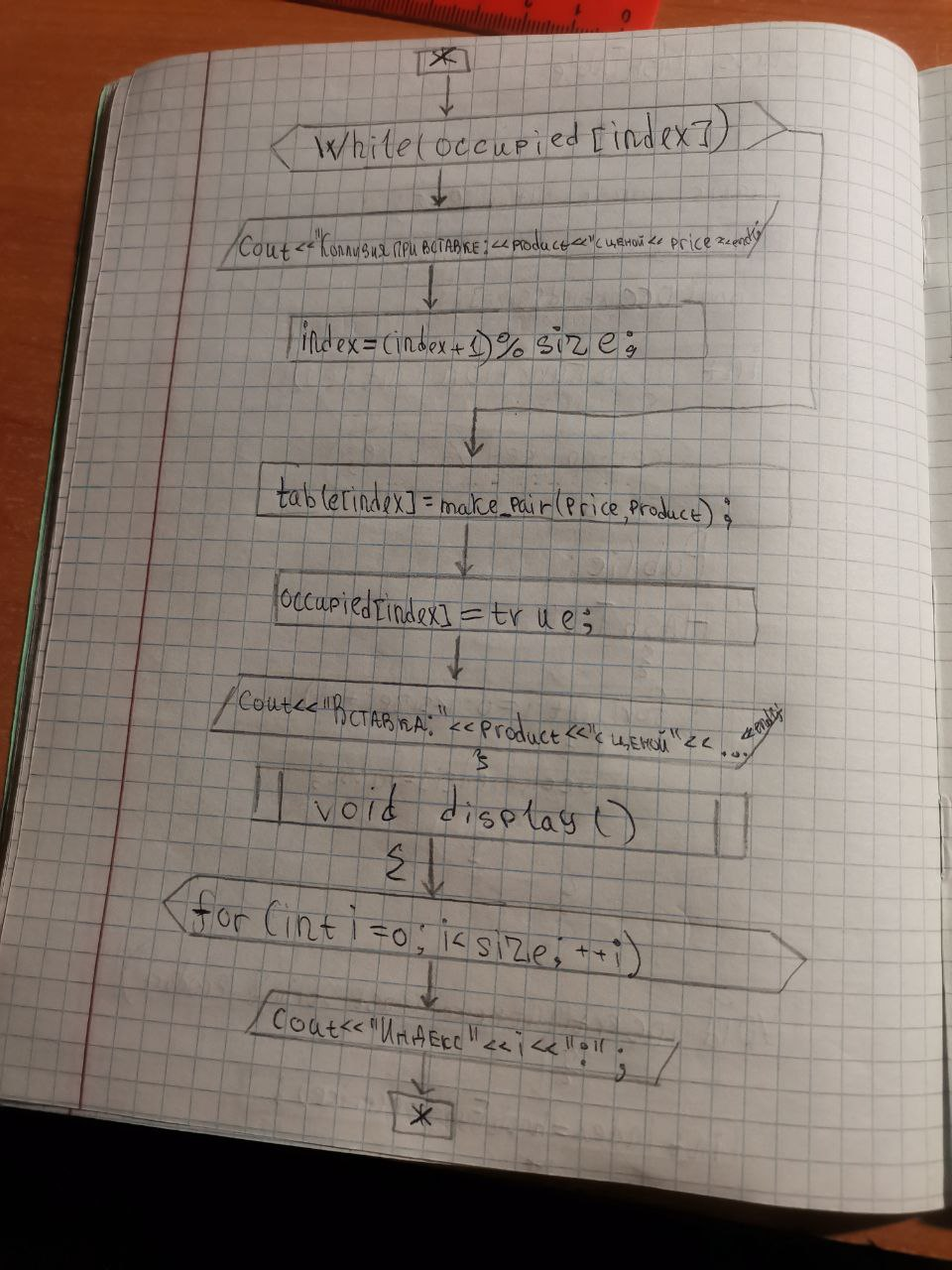
г. Пермь, 2024

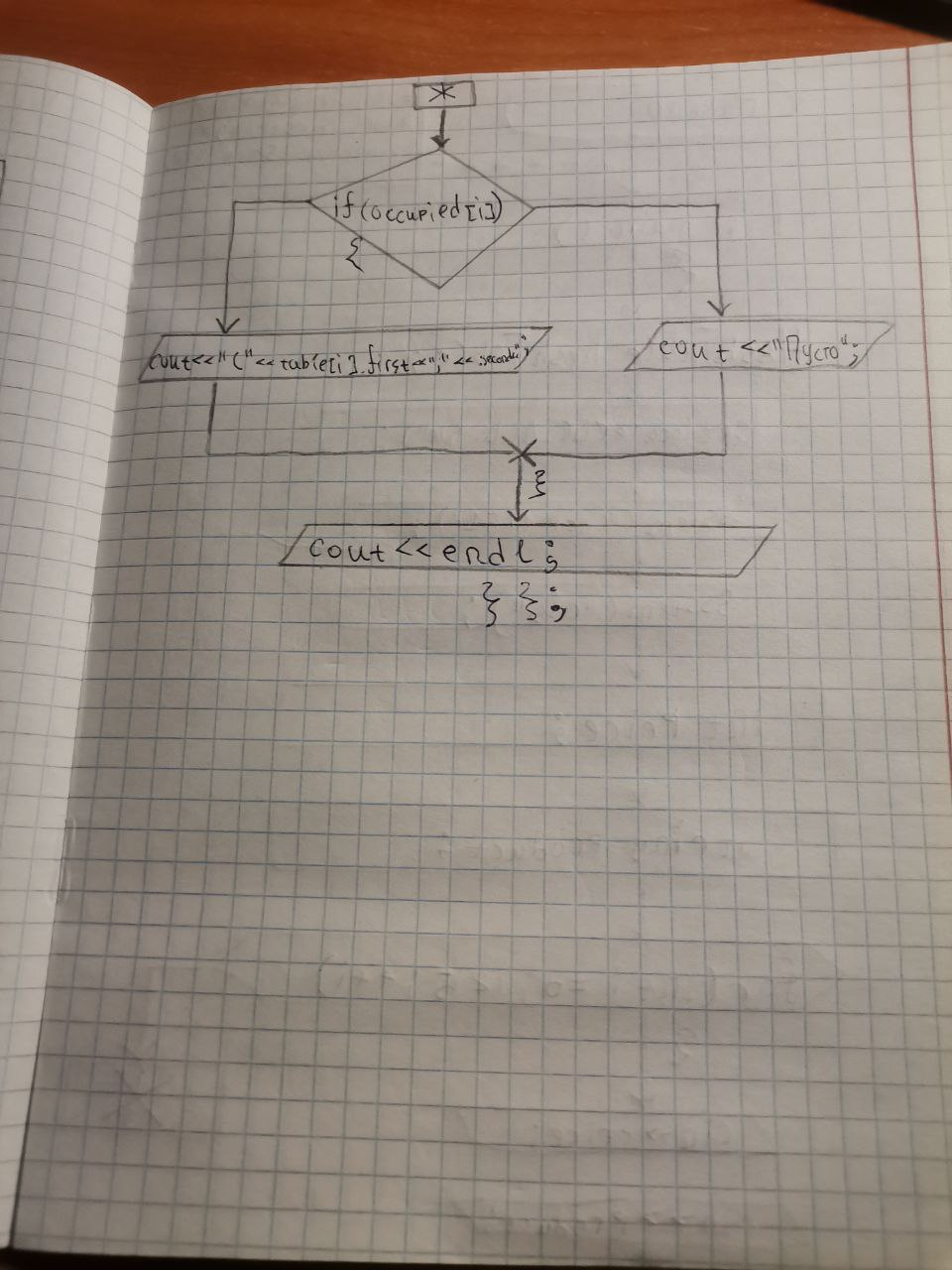
Постановка:

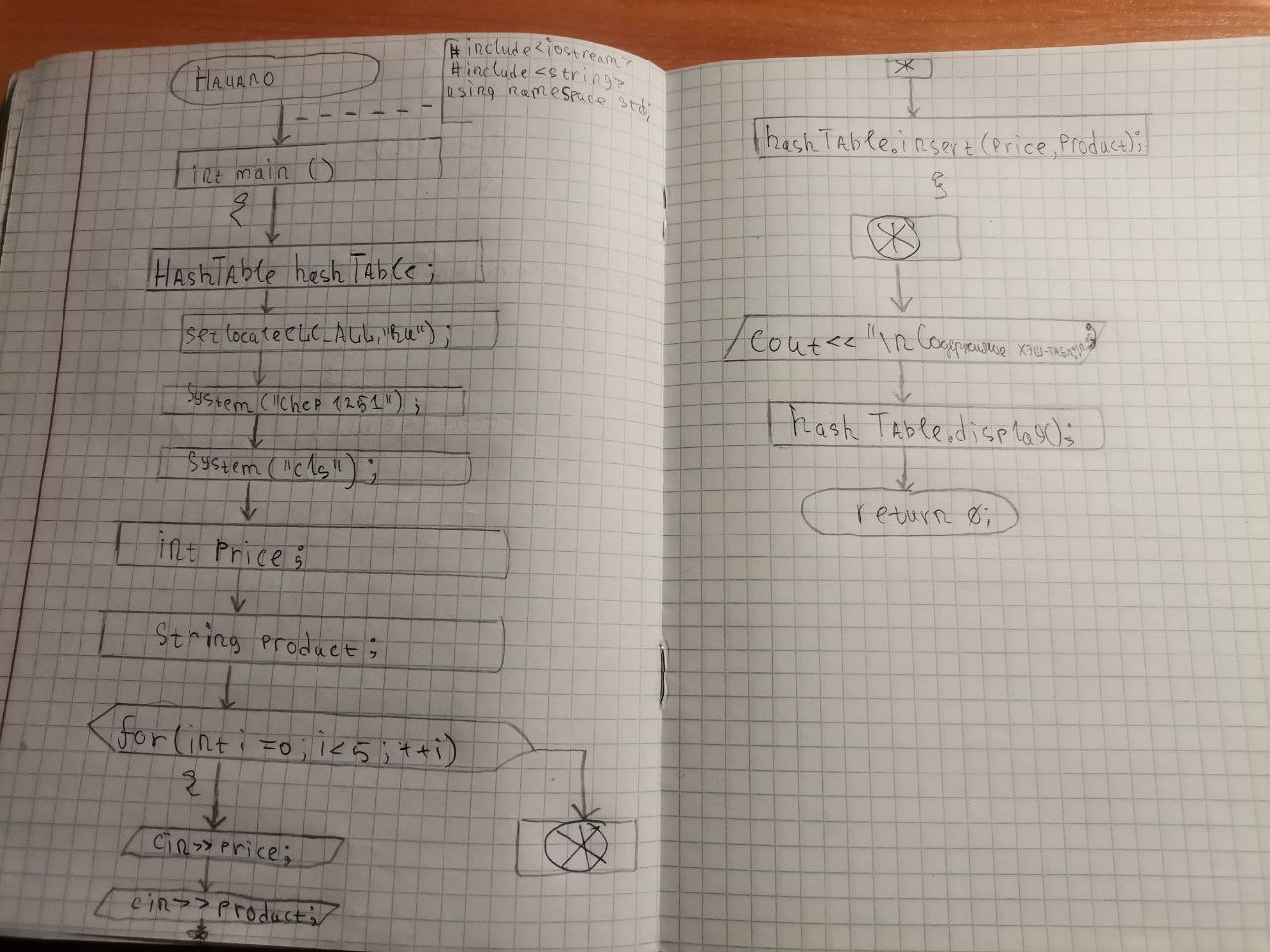
Создать хэш-таблицу и наглядно показать пример коллизий.

Блок-схема:









Код:

Линейный метод:  
#include <iostream>

#include <string>

using namespace std;

class HashTable

{

private:

static const int size = 5; // Размер хэш-таблицы

pair<int, string> table[size]; // Массив пар (цена, название продукта)

bool occupied[size]; // Массив для отслеживания занятости индексов

// Хэш-функция

int hashFunction(int key)

{

return key % size; // Простая хэш-функция

}

public:

HashTable()

{

// Инициализация массива занятости

for (int i = 0; i < size; ++i)

{

occupied[i] = false;

}

}

// Вставка элемента в хэш-таблицу

void insert(int price, const string& product)

{

int index = hashFunction(price);

// Проверка на наличие свободного места

while (occupied[index])

{

cout << "Коллизия при вставке: " << product << " с ценой " << price << " в индекс " << index << endl;

index = (index + 1) % size; // Переход к следующему индексу

}

table[index] = make\_pair(price, product); // Добавляем элемент в соответствующий индекс

occupied[index] = true; // Отмечаем индекс как занятый

cout << "Вставка: " << product << " с ценой " << price << " в индекс " << index << endl;

}

// Отображение хэш-таблицы

void display()

{

for (int i = 0; i < size; ++i)

{

cout << "Индекс " << i << ": ";

if (occupied[i])

{

cout << "(" << table[i].first << ", '" << table[i].second << "') ";

}

else

{

cout << "Пусто ";

}

cout << endl;

}

}

};

int main()

{

HashTable hashTable;

setlocale(LC\_ALL, "RU");

system("chcp 1251");

system("cls");

int price;

string product;

// Ввод данных пользователем

for (int i = 0; i < 5; ++i)

{ // Даем возможность ввести 5 элементов

cout << "Введите цену продукта: ";

cin >> price;

cout << "Введите название продукта: ";

cin >> product;

hashTable.insert(price, product);

}

// Отображение хэш-таблицы

cout << "\nСодержимое хэш-таблицы:\n";

hashTable.display();

return 0;

}

Метод цеочек:

#include <iostream>

#include <list>

#include <string>

using namespace std;

class HashTable

{

private:

static const int size = 5; // Размер хэш-таблицы

list<pair<int, string>> table[size]; // Массив списков для цепочек

// Хэш-функция

int hashFunction(int key)

{

return key % size; // Простая хэш-функция

}

public:

// Вставка элемента в хэш-таблицу

void insert(int price, const string& product)

{

int index = hashFunction(price);

// Добавляем элемент в соответствующий индекс (цепочку)

table[index].emplace\_back(price, product);

cout << "Вставка: " << product << " с ценой " << price << " в индекс " << index << endl;

}

// Отображение хэш-таблицы

void display()

{

for (int i = 0; i < size; ++i)

{

cout << "Индекс " << i << ": ";

for (const auto& item : table[i])

{

cout << "(" << item.first << ", '" << item.second << "') ";

}

if (table[i].empty())

{

cout << "Пусто ";

}

cout << endl;

}

}

};

int main()

{

HashTable hashTable;

setlocale(LC\_ALL, "RU");

system("chcp 1251");

system("cls");

int price;

string product;

// Ввод данных пользователем

for (int i = 0; i < 5; ++i) { // Даем возможность ввести 5 элементов

cout << "Введите цену продукта: ";

cin >> price;

cout << "Введите название продукта: ";

cin >> product;

hashTable.insert(price, product);

}

// Отображение хэш-таблицы

cout << "\nСодержимое хэш-таблицы:\n";

hashTable.display();

return 0;

}

Вывод:

