|  |
| --- |
| Федеральное государственное автономное образовательное учреждение высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  *Факультет компьютерных наук* |
| Лобастова Марина Антоновна  **ЛАБОРАТОРНАЯ РАБОТА № 5.33**  *Лабораторная работа*  студента образовательной программы «Программная инженерия» по направлению подготовки *09.03.04 Программная инженерия* |

Москва, 2021 год

Оглавление

[Задание «Дешифровка кодового сообщения» 3](#_Toc90500785)

[1 Постановка задачи 3](#_Toc90500786)

[2 Входные и выходные данные 3](#_Toc90500787)

[3 Алгоритм решения 3](#_Toc90500788)

[4 Программа для решения задачи 4](#_Toc90500789)

[Список использованной литературы 5](#_Toc90500790)

[ПРИЛОЖЕНИЕ А Программа класса Converter.h 6](#_Toc90500791)

[ПРИЛОЖЕНИЕ B Программа класса Converter.cpp 7](#_Toc90500792)

[ПРИЛОЖЕНИЕ C Программа класса homework5.cpp 8](#_Toc90500793)

Задание «Дешифровка кодового сообщения»

1. Постановка задачи

Узнав о планах преступников, Шерлок Холмс предложил лондонской полиции специальную машину для дешифровки сообщений злоумышленников. В качестве стандарта были выбраны «пляшущие человечки», шифр, в котором каждой букве латинского алфавита соответствует хитроумный значок. Реализовать многопоточное приложение, дешифрующее кодированный текст. В качестве ключа используется известная кодовая таблица, устанавливающая однозначное соответствие между каждой буквой и каким-нибудь числом. Процессом узнавания кода в решении задачи пренебречь. Каждый поток дешифрует свои кусочки текста. При решении использовать парадигму портфеля задач.

1. Входные и выходные данные

Входные данные представляют собой закодированное текстовое сообщение из малых латинских букв, каждой из которых присвоен свой индивидуальный двузначный численный код. Соответствие букв и их кодов представлено в Таблице

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **буква** | **значение** | **буква** | **значение** | **буква** | **значение** | **буква** | **значение** | **буква** | **значение** |
| a | 20 | g | 40 | m | 22 | s | 24 | y | 30 |
| b | 12 | h | 27 | n | 11 | t | 15 | z | 23 |
| c | 49 | i | 37 | o | 28 | u | 16 |  |  |
| d | 43 | j | 17 | p | 25 | v | 26 |  |  |
| e | 33 | k | 38 | q | 13 | w | 46 |  |  |
| f | 44 | l | 41 | r | 49 | x | 14 |  |  |

Таблица 1 – Соответствие букв латинского алфавита и их кодов

Если числовые значения не соответствуют зашифрованным латинским буквам, программа выведет сообщение об ошибке. Если данные корректны, дешифрованная символьная строка будет сохранена в файл «out.txt».

1. Алгоритм решения

Взаимодействующие равные – модель, в которой исключен не занимающийся непосредственными вычислениями управляющий поток. Распределение работ в таком приложении либо фиксировано заранее, либо динамически определяется во время выполнения. Одним из распространенных способов динамического распределения работ является «портфель задач». Портфель задач, как правило, реализуется с помощью разделяемой переменной, доступ к которой в один момент времени имеет только один процесс.

Метод портфеля задач заключается в поддержании очереди из задач. Каждый свободный поток берёт задачу из портфеля, выполняет её, при необходимости генерируя новые подзадачи и помещая их в портфель.

Портфель задач, как правило, реализуется с помощью разделяемой переменной, доступ к которой в один момент времени имеет только один процесс. Если же память ВС распределенная, то такая схема распределения работ превращается в схему «управляющий – рабочий», поскольку портфель задач оформляется отдельным процессом. Основными примерами в этой области являются научные вычисления с итеративными алгоритмами и системы, требующие децентрализованного принятия решений.

1. Программа для решения задачи

Программа для решения задания представлена в разделе «Приложения» (см. ПРИЛОЖЕНИЕ А, ПРИЛОЖЕНИЕ В, ПРИЛОЖЕНИЕ С).

Список использованной литературы

1. Kunle Olukotun «Chip Multiprocessor Architecture - Techniques to Improve Throughput and Latency»  Morgan and Claypool Publishers, 2007
2. Воронкин Р.А. «Технологии и методы программирования. Учебное пособие (практикум)». Ставрополь, 2014
3. Mario Nemirovsky, Dean M. Tullsen «Multithreading Architecture». Morgan and Claypool Publishers, 2013
4. Введение в технологии параллельного программирования // INTEL: сайт. URL: <https://software.intel.com/ru-ru/articles/writing-parallel-programs-a-multi-language-tutorial-introduction/> (дата обращения: 08.05.2017)

ПРИЛОЖЕНИЕ А **Программа класса Converter.h**

#ifndef HOMEWORK\_5\_CONVERTER\_H

#define HOMEWORK\_5\_CONVERTER\_H

#include <map>

#include <string>

using namespace std;

class Converter {

public:

map <int, std::string> code\_dictionary = { { 20, "a"}, {12, "b"}, {49, "c"},

{ 43, "d"}, { 33, "e"}, { 44, "f"}, { 40, "g"}, { 27, "h"},

{ 37, "i"}, { 17, "j"}, { 38, "k"}, { 41, "l"}, { 22, "m"},

{ 11, "n"}, { 28, "o"}, { 25, "p"}, { 13, "q"}, { 49, "r"},

{ 24, "s"}, { 15, "t"}, { 16, "u"}, { 26, "v"}, { 46, "w"},

{ 14, "x"}, { 30, "y"}, { 23, "z"}, { 0, " "}};

bool ifKey(int input);

string intToString(int input);

};

#endif //HOMEWORK\_5\_CONVERTER\_H

ПРИЛОЖЕНИЕ B **Программа класса Converter.cpp**

#include "Converter.h"

#include <string>

bool Converter::ifKey(int input) {

if (code\_dictionary.find(input) == code\_dictionary.end()) {

return false;

}

return true;

}

std::string Converter::intToString(int input) {

int mod;

string to\_return = "";

while (input > 0) {

mod = input % 100;

to\_return = code\_dictionary[mod] + to\_return;

input /= 100;

}

return to\_return;

}

ПРИЛОЖЕНИЕ C **Программа класса homework5.cpp**

#include <iostream>

#include <fstream>

#include <string>

#include <thread>

#include <mutex>

#include <chrono>

#include <vector>

#include "converter.h"

using namespace std;

static Converter conv;

void writeToFile(string\* result, int number) {

result->append(conv.intToString(number));

result->append(" ");

}

int main()

{

conv = Converter();

std::cout << "Enter a file name: ";

std::string filename;

std::cin >> filename;

std::ifstream in;

in.open(filename);

bool success = false;

while (!success) {

if (in.is\_open()) {

success = true;

}

if (!success) {

std::cout << "File does not exist. Please, try again: ";

std::cin >> filename;

in.open(filename);

}

}

string line;

int number, mod;

vector<int> tasks;

while (in >> line) {

try {

try {

number = std::stoi(line);

}

catch (std::logic\_error) {

throw exception();

}

if (line.size() % 2 != 0) {

throw exception();

}

while (number > 0) {

mod = number % 100;

number /= 100;

if (conv.code\_dictionary.find(mod) == conv.code\_dictionary.end()) throw exception();

}

tasks.push\_back(std::stoi(line));

tasks.push\_back(0);

}

catch (exception e) {

// interrupt all

in.close();

cout << "Invalid data! Conversion stopped.";

return 0;

}

}

in.close();

std::ofstream out;

out.open("out.txt");

string result = "";

std::mutex result\_mutex;

thread treads[10000];

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

{

result\_mutex.lock();

thread th(writeToFile, &result, tasks[i]);

treads[i] = std::move(th);

result\_mutex.unlock();

}

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

{

treads[i].join();

}

out << result;

std::cout << "File decryption finished successfully. Output file: " << "out.txt" << std::endl;

return 0;

}