

Федеральное государственное автономное образовательное
учреждение высшего образования
«Национальный исследовательский университет
«Высшая школа экономики»

Факультет компьютерных наук

Ахаладзе Мария Мерабиевна, БПИ193(2)

**ОТЧЕТ О ВЫПОЛНЕННОМ ДОМАШНЕМ ЗАДАНИИ ПО ДИСЦИПЛИНЕ «АРХИТЕКТУРА
ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ»**

Вариант 28

студента образовательной программы «Программная инженерия»
по направлению подготовки 09.03.04 Программная инженерия

Руководитель:

профессор департамента
программной инженерии
Легалов А. И.

Москва, 2020 год

Постановка задания

И снова пляшущие человечки. Узнав о планах преступников, озвученных в задаче 27, Шерлок Холмс предложил лондонской полиции специальную машину для дешифровки сообщений злоумышленников. 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.

Описание используемых моделей

Данное приложение было разработано в соответствии с моделью «Взаимодействующие равные», в частности, использовался способ «Портфель задач». Этот способ заключается в том, что к разделяемой переменной одновременно процессы не имеют доступ – только один процесс может иметь доступ в один момент времени; а так же все процессы имеют примерно одинаковый объем работ. Информация о модели взята из [1] и [2].

Разработка данного приложения была построена на применении OpenMP – это открытый стандарт для распараллеливания программ, написанных на языках программирования C, C++ и Фортран. Использовались директивы `parallel` (с установкой количества потоков, равного 4) для распараллеливания и `critical` для поочередного суммирования значений `returnValue`. Так же для вычисления номера потока использовался метод `omp_get_thread_num()`. Информация о работе с OpenMP взята из [3].

В приложении каждый поток оперирует с переменной `work`, в которой хранится часть зашифрованной строки, которую должен этот поток обработать. Причем нет ситуации доступа к общим данным, т.к. в каждом потоке значение `work` свое благодаря применению директивы `parallel`, т.е. невозможна ситуация доступа к одной и той же части строки. При этом перед

обработкой строки в цикле поочередно каждому потоку прибавляется в работу одна буква, пока все буквы не закончатся, т.е. у всех потоков примерно одинаковая часть работы.

Модель вычислений строится на том, чтобы разделять входную строку на приблизительно равные части (зависит от количества зашифрованных в ней букв), и каждую часть обрабатывать отдельным потоком. Шифры хранятся в `map`, где ключ – это некоторое число, а значение – это буква.

Для удобства все буквы кодируются одинаковым количеством цифр (четырьмя), а так же первая и последняя цифра – единицы, а вторая и третья могут быть любыми цифрами, кроме единицы, чтобы можно было легко и однозначно декодировать строку.

В блоке, следующем после директивы `parallel`, часть строки, которую обрабатывает поток, на каждой итерации выделяется подстрока из четырех цифр – очередная закодированная буква. Если полученное из строки число есть среди ключей таблицы шифра, то в дешифрованную строку добавляется значение, которое находится по ключу из таблицы, иначе значение `returnValue` в структуре, соответствующей данному потоку, равно единице. В блоке, следующем за директивой `critical`, поочередно потоки выполняют суммирование соответствующих им значений `returnValue`.

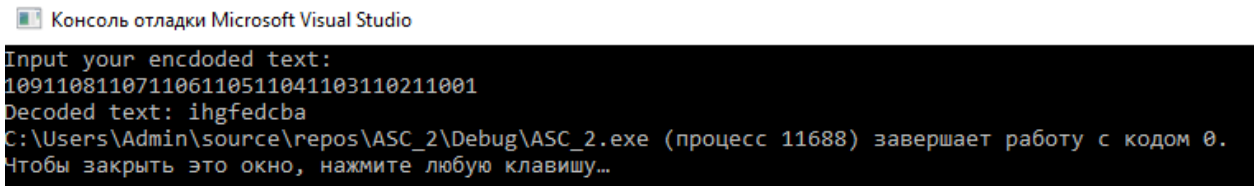
После блока параллельных потоков выполняется проверка суммы, записанной в `outputValues`. Если она равна нулю, то в цикле в результирующую строку по очереди записываются части расшифрованной строки, и выводится результирующая строка. В противном случае выводится сообщение, что строка имела некорректные данные, т.к. если сумма не равна нулю, то в строке есть шифр, который не соответствует таблице шифров.

Так же перед обработкой строки происходит проверка входной строки: если длина равна нулю или больше 100, если длина не кратна 4 (т.е. не все буквы закодированы четырьмя символами), если в строке есть не только цифры, то ввод считается некорректным и работа прекращается.

Листинг программы представлен в приложении А.

Тестирование программы

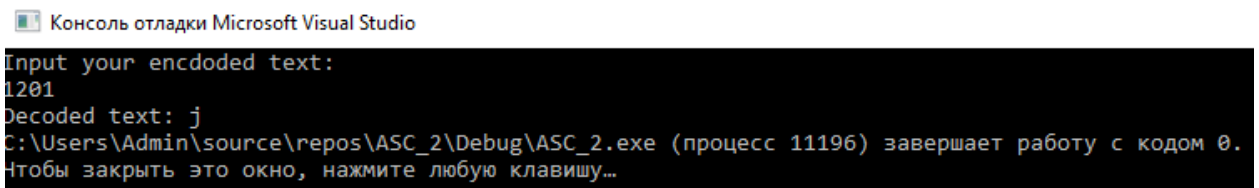
Результат тестирования случая, когда ввод корректный, длина является неграничным случаем (больше нуля, меньше 101) приведен на рисунке 1.



```
Консоль отладки Microsoft Visual Studio
Input your encoded text:
109110811071106110511041103110211001
Decoded text: ihgfedcba
C:\Users\Admin\source\repos\ASC_2\Debug\ASC_2.exe (процесс 11688) завершает работу с кодом 0.
Чтобы закрыть это окно, нажмите любую клавишу...
```

Рисунок 1 – Результат тестирования

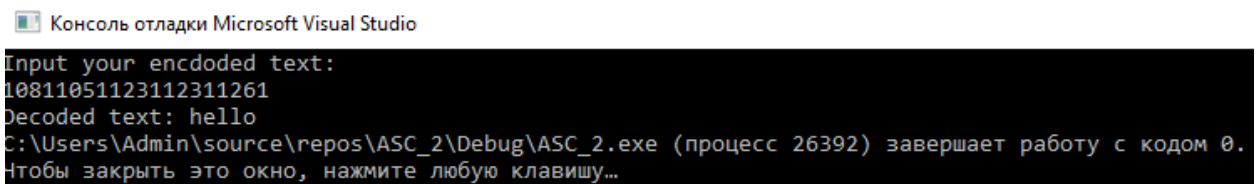
Результат тестирования случая, когда ввод корректный, длина не граничный случай, введена одна буква, приведен на рисунке 2.



```
Консоль отладки Microsoft Visual Studio
Input your encoded text:
1201
Decoded text: j
C:\Users\Admin\source\repos\ASC_2\Debug\ASC_2.exe (процесс 11196) завершает работу с кодом 0.
Чтобы закрыть это окно, нажмите любую клавишу...
```

Рисунок 2 – Результат тестирования

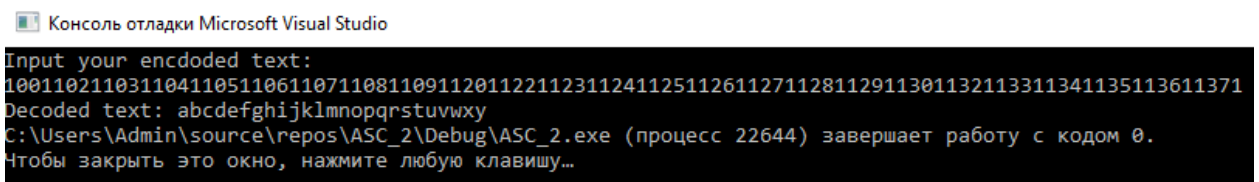
Результат тестирования случая, когда ввод корректный, длина не граничный случай, приведен на рисунке 3.



```
Консоль отладки Microsoft Visual Studio
Input your encoded text:
10811051123112311261
Decoded text: hello
C:\Users\Admin\source\repos\ASC_2\Debug\ASC_2.exe (процесс 26392) завершает работу с кодом 0.
Чтобы закрыть это окно, нажмите любую клавишу...
```

Рисунок 3 – Результат тестирования

Результат тестирования случая, когда ввод корректный, длина граничный случай (равна 100), приведен на рисунке 4.



```
Консоль отладки Microsoft Visual Studio
Input your encoded text:
1001102110311041105110611071108110911201122112311241125112611271128112911301132113311341135113611371
Decoded text: abcdefghijklmnopqrstuvwxyz
C:\Users\Admin\source\repos\ASC_2\Debug\ASC_2.exe (процесс 22644) завершает работу с кодом 0.
Чтобы закрыть это окно, нажмите любую клавишу...
```

Рисунок 4 – Результат тестирования

Результат тестирования случая, когда ввод некорректный (длина больше 100), приведен на рисунке 5.

Консоль отладки Microsoft Visual Studio

```
Input your encoded text:
1001102110311041105110611071108110911201122112311241125112611271128112911301132113311341135113611371181
Incorrect text length!
C:\Users\Admin\source\repos\ASC_2\Debug\ASC_2.exe (процесс 12848) завершает работу с кодом 0.
Чтобы закрыть это окно, нажмите любую клавишу...
```

Рисунок 5 – Результат тестирования

Результат тестирования случая, когда ввод некорректный (есть шифр, которого нет в таблице шифров), приведен на рисунке 6.

Консоль отладки Microsoft Visual Studio

```
Input your encoded text:
1001111110211991
Sorry but there are inappropriate values in your encoded text
C:\Users\Admin\source\repos\ASC_2\Debug\ASC_2.exe (процесс 24804) завершает работу с кодом 0.
Чтобы закрыть это окно, нажмите любую клавишу...
```

Рисунок 6 – Результат тестирования

Результат тестирования случая, когда ввод некорректный (во входной строке не только цифры), приведен на рисунке 7.

Консоль отладки Microsoft Visual Studio

```
Input your encoded text:
1!!1
Sorry but there are incorrect symbols in your encoded text
C:\Users\Admin\source\repos\ASC_2\Debug\ASC_2.exe (процесс 25252) завершает работу с кодом 0.
Чтобы закрыть это окно, нажмите любую клавишу...
```

Рисунок 7 – Результат тестирования

Результат тестирования случая, когда ввод некорректный (длина не кратна 4), приведен на рисунке 8.

Консоль отладки Microsoft Visual Studio

```
Input your encoded text:
101121
Incorrect text length!
C:\Users\Admin\source\repos\ASC_2\Debug\ASC_2.exe (процесс 20116) завершает работу с кодом 0.
Чтобы закрыть это окно, нажмите любую клавишу...
```

Рисунок 8 – Результат тестирования

Библиографический список

[1] АЛГОРИТМЫ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ И ПРОГРАММИРОВАНИЕ: КУРС ЛЕКЦИЙ // Единое окно доступа к образовательным ресурсам URL: http://window.edu.ru/catalog/pdf2txt/971/67971/41350?p_page=20 (дата обращения: 12.12.2020).

[2] Основы многопоточного и распределенного программирования // Визуальное программирование и MFC URL: <http://www.soft.architecture.net.ru/70/index-upravljajushhij-rabochie-raspredeleennyj-portfel.htm> (дата обращения: 12.12.2020).

[3] Основные директивы OpenMP с примерами // Чертоги разума. Личный блог Кузьминых Кирилла URL: <http://mindhalls.ru/pragma-omp-directives-samples/> (дата обращения: 12.12.2020).

Приложение А

Листинг программы

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <iostream>
#include <thread>
#include <map>
#include <string>

// Метод создания map
std::map<int, char> makeMap() {
    std::map<int, char> codeTable;
    codeTable.insert(std::make_pair(1001, 'a'));
    codeTable.insert(std::make_pair(1021, 'b'));
    codeTable.insert(std::make_pair(1031, 'c'));
    codeTable.insert(std::make_pair(1041, 'd'));
    codeTable.insert(std::make_pair(1051, 'e'));
    codeTable.insert(std::make_pair(1061, 'f'));
    codeTable.insert(std::make_pair(1071, 'g'));
    codeTable.insert(std::make_pair(1081, 'h'));
    codeTable.insert(std::make_pair(1091, 'i'));
    codeTable.insert(std::make_pair(1201, 'j'));
    codeTable.insert(std::make_pair(1221, 'k'));
    codeTable.insert(std::make_pair(1231, 'l'));
    codeTable.insert(std::make_pair(1241, 'm'));
    codeTable.insert(std::make_pair(1251, 'n'));
    codeTable.insert(std::make_pair(1261, 'o'));
    codeTable.insert(std::make_pair(1271, 'p'));
    codeTable.insert(std::make_pair(1281, 'q'));
    codeTable.insert(std::make_pair(1291, 'r'));
    codeTable.insert(std::make_pair(1301, 's'));
    codeTable.insert(std::make_pair(1321, 't'));
    codeTable.insert(std::make_pair(1331, 'u'));
    codeTable.insert(std::make_pair(1341, 'v'));
    codeTable.insert(std::make_pair(1351, 'w'));
    codeTable.insert(std::make_pair(1361, 'x'));
    codeTable.insert(std::make_pair(1371, 'y'));
    codeTable.insert(std::make_pair(1381, 'z'));
    return codeTable;
}

const int threadsNumber = 4;          // Количество потоков
const std::map<int, char> codeTable = makeMap();    // Таблица шифров

typedef struct threadData { // Структура для хранения данных, полученных в работе потока
    std::string res;        // Дешифрованная строка
    std::string work;       // Зашифованная строка
    int returnValue;        // Число, характеризующее, верна ли входная зашифованная
    строка
};

void Decoding(threadData &data) { // Метод расшифровки
    std::string work = data.work;
    for (int i = 0; i < work.length() / 4; i++) { // Делим длину на четыре, т.к. все
        буквы закодированы четырьмя цифрами
        int encodedLetter = std::stoi(work.substr(i * 4, 4)); // Парсинг строки
        if (codeTable.count(encodedLetter) > 0) { // Если такое число есть в
            таблице шифров

```

```

        char decodedLetter = codeTable.at(encodedLetter);    // Находим,
        // какая буква закодирована таким числом
        data.res += decodedLetter; // Записываем букву в дешифрованную строку
    }
    else data.returnValue = 1; // Возвращаем 1, т.к. входные данные некорректны
}
}
bool IsDigitString(const std::string &text)    // Проверка, состоит ли строка только из
// чисел
{
    return text.find_first_not_of("0123456789") == std::string::npos;    // Проверка,
    // что нет символа, не обозначающего цифру
}
int LetterIndex(int workSize[], int index) {    // Получение индекса текущей буквы
    int sum = 0;
    for (int i = 0; i < index; i++) {
        sum += workSize[i]; // Суммируем индексы всех предыдущих букв
    }
    return sum;
}

int main() {

    std::string text;
    std::cout << "Input your encoded text: \n";
    std::cin >> text; // Ввод дешифрованной строки

    if (text.length() % threadsNumber != 0 || text.length() == 0 || text.length() >
100) {    // Если количество символов не кратно 4 (все буквы кодируются 4 цифрами)
    // или строка пустая или длина больше 100, это некорректный ввод
        std::cout << "Incorrect text length!";
        return 0;
    }
    if (!IsDigitString(text)) { // Если в строке есть не только цифры, это некорректный
    // ввод (буквы кодируются только числовыми последовательностями)
        std::cout << "Sorry but there are incorrect symbols in your encoded text";
        return 0;
    }

    int letterCount = text.length() / 4;    // Т.к. буквы кодируются 4 цифрами, то
    // количество букв в четыре раза меньше количества символов
    int workSize[threadsNumber] = { 0 };    // Массив, в котором будет храниться
    // количество букв, которые нужно дешифровать, для каждого потока
    for (int i = 0; i < letterCount; i++) { // Раздаем каждому потоку некоторое
    // количество букв
        workSize[i % 4]++;    // Поочередно каждому потоку даем по еще одной букве,
        // пока цикл не закончится
    }

    threadData threadsData[threadsNumber]; // Массив структур, хранящих данные о
    // работе потоков
    int outputValues = 0; // Сумма чисел, характеризующих корректные данные в частях
    // зашифрованной строки

#pragma omp parallel num_threads(threadsNumber) // Распараллеливание на четыре потока
    {
        auto i = omp_get_thread_num();    // Получаем номер потока
        std::string work = text.substr(LetterIndex(workSize, i) * 4, workSize[i] *
4); // Определяем первую букву для i-того потока и умножаем количество букв на 4 (равно
        // количеству символов, которые надо обработать)
        threadsData[i].res = "";    // Присваиваем начальные данные для
        // дешифрованной строки
        threadsData[i].work = work; // Присваиваем начальные данные для
        // зашифрованной строки
    }
}

```



```

        threadsData[i].returnValue = 0;    // Присваиваем начальные данные для числа,
показывающего, корректные ли данные в зашифрованной строке
        Decoding(std::ref(threadsData[i]));    // Вызываем в i-том потоке метод
дешифровки
#pragma omp critical // Критическая секция, блок выполняется по очереди всеми потоками
        {
            outputValues += threadsData[i].returnValue;    // Суммируем
returnValue каждого потока
        }
    }
    if (outputValues != 0) std::cout << "Sorry but there are inappropriate values in
your encoded text"; // Если сумма не равна нулю, то хотя бы в одной части зашифрованной
строки есть некорректные символы
    else
    {
        std::string decodedText;    // Результирующая строка
        for (int i = 0; i < threadsNumber; i++)
            decodedText += threadsData[i].res;    // Записываем в
результирующую строку res
        std::cout << "Decoded text: " + decodedText;    // Вывод расшифрованной
строки
    }
}

```