ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа бакалавриата «Программная инженерия»

ЗАДАНИЕ 4

Пояснительная записка

Выполнил студент группы БПИ196 (2)

Шестаков Михаил Сергеевич

Вариант 28. И снова пляшущие человечки. Узнав о планах преступников, 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.

М о с к в а 29.11.2020

1. Модель вычислений

В программе используется парадигма портфеля задач [1], [2]. Программа создаёт несколько потоков с помощью #pragma omp parallel [3]. Затем массив, который нужно дешифровать разбивается на несколько равных частей и с помощью #pragma omp task [4] каждая из частей складывается в портфель задач. Затем потоки начинают выполнение задач, синхронизация окончания выполнения осуществляется с помощью команды #pragma omp taskwait. [4]

После выполнения очередной задачи поток выводит об этом информацию в консоль. Для того, чтобы в один момент времени выводить в консоль мог только один поток используется критическая секция -- #pragma omp critical. [5]

2. Текст программы

Для удобства текст программы разбит на несколько файлов

2.1 main.c

```
#include <stdio.h>
#include <pthread.h>
#include <stdbool.h>
#include <stdlib.h>
#include <omp.h>
#include "Util.h"
#include "Cipher.h"


#define countPerTask 10000


// структура для хранения результата выполнения потока
struct ThreadResult {
    bool success;
    int wrongValue;
};


struct ThreadResult result = {.success=true};

int* encoded; // закодированные данные -- числа
char* decoded; // декодированные данные -- символы (c-string)


// выходит из потока с ошибкой
void reportError(int value) {
    result.success = false;
    result.wrongValue = value;
}

// создаёт задачи
void createTasks(int count) {
    for (int i = 0; i < count; i += countPerTask) {
        int index = i;
```

```
        #pragma omp task
        {
            int firstIndex = index;
            int lastIndex = min(index + countPerTask, count);
            decode(encoded, decoded, firstIndex, lastIndex);

            int numberOfThread = omp_get_thread_num();
            int countOfThreads = omp_get_num_threads();
            #pragma omp critical
            {
                printf("Thread %d / %d finished task %d\n", numberOfThread, countOfThreads,
index);
            }
        }
    }
}


int main(int argc, char** argv) {
    // количество символов
    int count;
    // считываем исходные данные
    if (!readCipherData(&count, &encoded, &decoded)) {
        pause();
        return 1;
    }

    // запускаем секцию с несколькими потоками
    #pragma omp parallel
    {
        //  в одном потоке создаём задачи
        #pragma omp single nowait
        {
            createTasks(count);
        }
        // ждём, пока все задачи завершатся
        #pragma omp taskwait
    }

    // проверяем флаг ошибки
    if (!result.success) {
        printf("Invalid value: %d\n", result.wrongValue);
        pause();
        return -1;
    }


    printf("Decoded message: ");
    // выводим результат
    puts(decoded);

    // ждём ввода пользователя
    pause();


    // очищаем память
    free(encoded);
    free(decoded);

    return 0;
}
```

2.2 Cipher.h

```
#ifndef CIPHER_H
#define CIPHER_H

#include <stdio.h>
#include <stdlib.h>

char cipherTable[255];
void reportError(int value);

// расшифровывает данные с промежутка [firstIndex, lastIndex)
void decode(int* encoded, char* decoded, int firstIndex, int lastIndex) {
    for (int i = firstIndex; i < lastIndex; i++) {
        int value = encoded[i];
        if (value > 256 || value < 0 || cipherTable[value] == 0) {
            reportError(value);
        }
        decoded[i] = cipherTable[value];
    }
}

// читает таблицу шифрования
bool readCipherTable() {
    int cipherCount = 0;
    printf("Count of cipher entries: ");
    if (scanf("%d", &cipherCount) != 1) {
        printf("Unable to read count of cipher entries\n");
        return false;
    }
    if (cipherCount < 0 || cipherCount > 256) {
        printf("Incorrect count of cipher entries\n");
        return false;
    }

    char tmp[100];
    printf("Cipher entries (in format '<character> = <number>\\n'):\n");
    for (int i = 0; i < cipherCount; i++) {
        char c;
        int number;
        scanf("%[\n ]", tmp); // пропускаем пробелы и переносы строк
        if (scanf("%c%[= ]", &c, tmp) != 2 || scanf("%d", &number) != 1) {
            printf("Unable to read cipher entry\n");
            return false;
        }
        if (number < 0 || number > 255) {
            printf("Incorrect number: %d. Number should be between 0 and 255\n", number);
            return false;
        }
        if (cipherTable[number] != 0) {
            printf("Repeating entries in cipher table");
            return false;
        }
        cipherTable[number] = c;
    }

    return true;
}
```

```
// читает данные, которые нужно расшифровать
bool readEncoded(int count, int* encoded) {
    printf("Encoded characters: (in format '<number1> <number2> ... <numberN>'\n");
    for (int i = 0; i < count; i++) {
        if (scanf("%d", encoded + i) != 1) {
            printf("Unable to read %d encoded\n", i + 1);
            return false;
        }
    }

    return true;
}

// читает данные, которые нужно расшифровать и таблицу шифрования
bool readCipherData(int* count, int** encoded, char** decoded) {
    printf("Count of encoded characters: ");
    if (scanf("%d", count) != 1) {
        printf("Unable to read encoded count\n");
        return false;
    }
    if (*count < 0 || *count > 5e8) {
        printf("Incorrect encoded count\n");
        return false;
    }

    *encoded = malloc(*count * sizeof(int));
    *decoded = malloc((*count + 1) * sizeof(char));
    (*decoded)[*count] = 0; // записываем в конец нуль-турминатор

    if (!readEncoded(*count, *encoded)) {
        return false;
    }

    return readCipherTable();
}

#endif
```

2.3 Util.h

```
#ifndef UTIL_H
#define UTIL_H

// ждёт от пользователя ввода
void pause() {
#ifdef _WIN32
    system("pause");
#else
    system("read");
#endif
}

// находит минимальное число из двух
int min(int x, int y) {
    if (x < y) {
        return x;
    }
    return y;
}

#endif
```

3. Тестирование

3.1. Простой тест

```
[user@laptopmishash AVS_task_openmp]$ ./program
Count of encoded characters: 3
Encoded characters: (in format '<number1> <number2> ... <numberN>'
3 1 2
Count of cipher entries: 3
Cipher entries (in format '<character> = <number>\n'):
a = 3
b = 2
c = 1
Thread 3 / 8 finished task 0
Decoded message: acb
Press enter to continue...
```

3.2. Пустой тест

```
[user@laptopmishash AVS_task_openmp]$ ./program
Count of encoded characters: 0
Encoded characters: (in format '<number1> <number2> ... <numberN>'
Count of cipher entries: 0
Cipher entries (in format '<character> = <number>\n'):
Decoded message:
Press enter to continue...
```

3.3. Большой тест (1 000 000 символов)

test1.txt

```
1000000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 1 2 3 4 5
26
a = 1
b = 2
c = 3
d = 4
e = 5
f = 6
g = 7
h = 8
i = 9
j = 10
k = 11
l = 12
m = 13
n = 14
o = 15
p = 16
q = 17
r = 18
s = 19
t = 20
u = 21
v = 22
w = 23
x = 24
y = 25
z = 26
```

```
[user@laptopmishash AVS_task_openmp]$ ./program < tests/test1.txt > tests
/output1.txt
```

```
Thread 1 / 8 finished task 0
Thread 6 / 8 finished task 10000
Thread 0 / 8 finished task 30000
Thread 5 / 8 finished task 20000
Thread 6 / 8 finished task 50000
Thread 1 / 8 finished task 40000
Thread 0 / 8 finished task 60000
Thread 5 / 8 finished task 70000
Thread 6 / 8 finished task 80000
Thread 1 / 8 finished task 90000
Thread 0 / 8 finished task 100000
Thread 5 / 8 finished task 110000
Thread 6 / 8 finished task 120000
Thread 1 / 8 finished task 130000
Thread 0 / 8 finished task 140000
Thread 5 / 8 finished task 150000
Thread 6 / 8 finished task 160000
Thread 1 / 8 finished task 170000
Thread 0 / 8 finished task 180000
Thread 5 / 8 finished task 190000
Thread 6 / 8 finished task 200000
```

```
Thread 5 / 8 finished task 990000
Decoded message: abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopq
rstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefgh
ijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxy
zabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnop
qrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefg
hijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwx
yzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmno
pqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdef
ghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvw
xyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmn
opqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcde
fghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuv
wxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklm
nopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcd
efghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstu
vwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijkl
mnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabc
defghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrst
uvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijk
lmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzab
```

3.4. Очень большой тест (1000 000 000 символов). Вместо входного массива случайные данные, вывод результата отключен.

```
[user@laptopmishash AVS_task_openmp]$ time ./program > result.txt

real    0m6.432s
user    0m10.878s
sys     0m2.203s
```

```
  GNU nano 5.2                    result.txt
Thread 5 / 8 finished task 100000
Thread 3 / 8 finished task 0
Thread 7 / 8 finished task 51300000
Thread 4 / 8 finished task 200000
Thread 5 / 8 finished task 300000
Thread 3 / 8 finished task 400000
Thread 7 / 8 finished task 51600000
Thread 4 / 8 finished task 500000
Thread 3 / 8 finished task 700000
Thread 6 / 8 finished task 800000
Thread 7 / 8 finished task 52000000
Thread 4 / 8 finished task 900000
Thread 3 / 8 finished task 1000000
Thread 6 / 8 finished task 1100000
                        [ Read 10001 lines ]
```

3.5. Все ASCII символы

```
94
33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 5
7 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81
 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 10
4 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122
 123 124 125 126
94
! = 33
" = 34
# = 35
$ = 36
% = 37
& = 38
' = 39
( = 40
) = 41
* = 42
+ = 43
, = 44
- = 45
"tests/test2.txt" 97L, 1001B                          6,1           Top
```

```
[user@laptopmishash AVS_task_openmp]$ ./program < tests/test2.txt > tests
/output2.txt
```

```
Count of encoded characters: Encoded characters: (in format '<number1> <n
umber2> ... <numberN>'
Count of cipher entries: Cipher entries (in format '<character> = <number
>\n'):
Thread 7 / 8 finished task 0
Decoded message: !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWX
YZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
Press enter to continue...
"tests/output2.txt" [noeol] 5L, 343B                  5,1           All
```

3.6. Некорректные данные – количество символов

```
[user@laptopmishash AVS_task_openmp]$ ./program
Count of encoded characters: abab
Unable to read encoded count
Press enter to continue...
```

37. Некорректные данные – количество символов слишком маленькое

```
[user@laptopmishash AVS_task_openmp]$ ./program
Count of encoded characters: -300
Incorrect encoded count
Press enter to continue...
```

3.8. Некорректные данные – количество символов слишком большое

```
[user@laptopmishash AVS_task_openmp]$ ./program
Count of encoded characters: 1000000000
Incorrect encoded count
Press enter to continue...
```

3.9.Некорректные данные – несуществующий закодированный символ

```
[user@laptopmishash AVS_task_openmp]$ ./program
Count of encoded characters: 3
Encoded characters: (in format '<number1> <number2> ... <numberN>'
1 2 37
Count of cipher entries: 2
Cipher entries (in format '<character> = <number>\n'):
a = 1
b = 2
Thread 5 / 8 finished task 0
Invalid value: 37
Press enter to continue...
```

3.10. Некорректные данные – слишком мало символов в таблице шифрования

```
[user@laptopmishash AVS_task_openmp]$ ./program
Count of encoded characters: 1
Encoded characters: (in format '<number1> <number2> ... <numberN>'
10
Count of cipher entries: -2
Incorrect count of cipher entries
Press enter to continue...
```

3.11. Некорректные данные – слишком много символов в таблице шифрования

```
[user@laptopmishash AVS_task_openmp]$ ./program
Count of encoded characters: 1
Encoded characters: (in format '<number1> <number2> ... <numberN>'
10
Count of cipher entries: 10000
Incorrect count of cipher entries
Press enter to continue...
```

3.12. Некорректные данные – неправильный формат в таблице шифрования

```
[user@laptopmishash AVS_task_openmp]$ ./program
Count of encoded characters: 1
Encoded characters: (in format '<number1> <number2> ... <numberN>'
10
Count of cipher entries: 3
Cipher entries (in format '<character> = <number>\n'):
test
Unable to read cipher entry
Press enter to continue...
```

Источники

[1] https://l.wzm.me/_coder/custom/parallel.programming/003.htm

[2] http://softcraft.ru/edu/comparch/tasks/t04/

[3] https://docs.microsoft.com/en-us/cpp/parallel/openmp/reference/openmp-directives?view=msvc-160#parallel

[4] https://en.wikibooks.org/wiki/OpenMP/Tasks

[5] https://docs.microsoft.com/en-us/cpp/parallel/openmp/reference/openmp-directives?view=msvc-160#critical