ГУАП

КАФЕДРА № 53

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ассистент |  |  |  | С. Ю. Гуков |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| №4.2. РАЗРАБОТКА АЛГОРИТМА ИСЧЕРПЫВАЮЩЕГО ПОИСКА (ПОЛНОГО ПЕРЕБОРА) |
| Вариант 6 (15) |
| по курсу: ОСНОВЫ ПРОГРАММИРОВАНИЯ |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 5136 |  | , |  | Д. Л. Мягков |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

Оглавление:

[Цель. 3](#__RefHeading___Toc11772_844291809)

[Индивидуальное задание. 4](#__RefHeading___Toc11774_844291809)

[Описание алгоритма 5](#__RefHeading___Toc11776_844291809)

[Листинг программы. 6](#__RefHeading___Toc11778_844291809)

[Вывод. 25](#__RefHeading___Toc11780_844291809)

# Цель.

Реализовать алгоритм на языке C/С++, выполняющий поставленную задачу (АЛГОРИТМ ДОЛЖЕН БЫТЬ РЕАЛИЗОВАН ПУТЕМ ПЕРЕБОРА ВСЕХ ВОЗМОЖНЫХ ВАРИАНТОВ РЕШЕНИЯ). Вариант задания, пример входных и выходных данных представлен в таблице 9. Глобальные параметры использовать запрещено; допустимо использование дополнительных функций.

* Разработанный алгоритм должен быть реализован в виде цельной программной функции (или нескольких функций) так, чтобы мог быть многократно применим с различными исходными данными и при этом не включал команды, не относящиеся к решаемой задаче, например, ввод и вывод исходных данных на консоль или в файл.
* Произвести экспериментальную проверку времени работы разработанного алгоритма, определив его класс сложности для среднего случая. Измерить среднее время для Test\_Count повторений при различных размерностях входных данных. !ВАЖНО: при относительной небольшой размерности входных данных программа может потребовать для выполнения значительное время, рекомендуется не ждать выполнения программы в случае превышения времени выполнения 15 минут.

# Индивидуальное задание.

## 

Рисунок 1. Индивидуальное задание

# Описание алгоритма

Алгоритм представлен функцией combinations. Она принимает количество операций count, предыдущее значение last, массив индексов indexes, массив объектов skills\_and\_salaries, исходный массив пар P.

1. В ней создается пустое множество unification, сумма зарплат sum\_salaries.

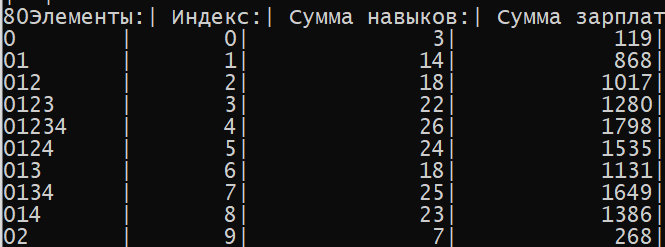
Потом проходит по массиву индексов и копирует каждое множество из каждоый пары соответсвующего индекса в множество unification и к сумме зарплат прибавляет соответствующую зарплату. Увеличивает количество операций на 1.

Потом, если длина массива индексов не равна нулю, то мы создаем объект класса Skills\_and\_salaries sk\_and\_sa, индексу суммы присваивается размер массива skills\_and\_salaries, вызывается метод класса, чтобы полям объекта sk\_and\_sa, присвоить массив индексов, сумму зарплат и множество unification (сумма навыков). В массив skills\_and\_salaries добавляется объект.

Потом, запускается цикл, с index = last + 1, пока index <= P.size(). В нем в массив индексов добавляется index – 1, то есть сначала добавляется 0. Потом вызывается снова вызывается функция combinations, но в нее уже передается на место элемента last элемент index.

2. То есть мы будем добавлять в массив индексов индексы пока last + 1 меньше или равен длины массива P. last каждый вызов рекурсии увеличивается на 1.

Когда last + 1 становится больше размера P, мы не заходим в цикл и выходим из фукнции. После выхода из функции мы возвращаемся на предыдущий вызов функции и удаляем последний индекс из массива индексов. Index увеличивается на 1 и вновь превышает размер P. Значит в цикл мы не войдем и выйдем снова из функции и вернемся на предыдущий шаг рекурсии. Потом снова произойдут действия из пункта 1 и мы зайдем в цикл, но начальное значение будет равно предыдущему last + 1.



То есть от 0 до 4 мы зашли 5 раз в функцию, потом вышли из последней функции, last будет равен 3, значит index будет равен 4, и получается что мы пропустим 3, зайдем один раз в цикл и снова выйдем из функции. Last будет равен 2, index будет равен 3, значит мы зайдем в цикл два раза, сначала когда index будет равен 3, потом когда index будет равен 4. И так мы будем идти пока не придем к первому вызову рекурсии, и index не станет равен P.

Таким образом мы пройдемся по всем сочетаниям персонажей.

# Листинг программы.

Листинг lab\_4\_2.cpp:

#include <iostream>// библиотека

#include <iomanip>

#include <Windows.h> // библиотека

#include "various\_functions.h"

#include "lab\_4\_2\_functions.h"

#include "main\_header.h"

#include "string"

#include "vector"

#include "Skills\_and\_salaries.h"

#include <set>

#include <iterator>

using namespace std; //подключение пространства имен std

void lab\_4\_2()

{

setlocale(0, ""); // поддержка кириллицы в консоли (вывод)

SetConsoleCP(1251); // поддержка кириллицы в консоли (ввод)

SetConsoleOutputCP(1251); // поддержка кириллицы в консоли (ввод)

unsigned int N, size;

bool mode, print;

double average\_time = 0;

cout << "Ручной режим (1) или автоматический (0): ";

mode = get\_number<int>();

if (mode == 1)

{

cout << "Введите количество измерений: ";

N = get\_number<int>();

cout << "Введите количество человек: ";

size = get\_number<int>();

cout << "Печатать? Да(1), нет(0): ";

print = get\_number<bool>();

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

{

average\_time += find\_time(size, print);

}

cout << "среднее время для " << N << " измерений" << " = " << average\_time / N << "мc" << ", кол-во человек = " << size;

}

else

{

N = 1000;

for (int j = 2; j < 12; j++)

{

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

{

average\_time += find\_time(j, 0);

}

cout << "среднее время для " << N << " измерений" << " = " << setw(9) << average\_time / N << " мc" << ", кол-во человек = " << j << endl;

}

}

}

Листинг lab\_4\_2\_functions.cpp:

#include <iostream>// библиотека

#include <iomanip>

#include <Windows.h> // библиотека

#include "various\_functions.h"

#include "lab\_4\_2\_functions.h"

#include "string"

#include "vector"

#include "Skills\_and\_salaries.h"

#include <set>

#include <iterator>

using namespace std; //подключение пространства имен std

int find\_time(unsigned int size, bool print)

{

int start\_time, end\_time;

srand(clock());

// N = количество претендетов

// lS = количество навыков у претендента

// lS\_max = максимальное количество навыков

// M\_max = общее максимальное количество неповторяющихся навыков

// Z\_max = бюджет

bool debugging = 1;

//unsigned short N = input\_count\_people();

unsigned short N = size;

//unsigned short lS\_max = in\_count\_ckills();

unsigned short lS\_max = 30;

unsigned short Z\_max = N \* (lS\_max \* 8) + rand() % 10;

vector < pair < set <int>, int> > P(N);

set <set<int>> sets\_indexes;

vector < Skills\_and\_salaries > skills\_and\_salaries;

vector <int> indexes = {};

fill\_pair(P, N, lS\_max, Z\_max);

unsigned short l\_void = sqrt(N) + 1;

if (print == 1)

{

cout << "Бюджет = " << setw(l\_void) << Z\_max << endl;

print\_three\_vector(P, lS\_max);

}

int count = 0;

int time\_start = clock();

combinations(count, 0, indexes, skills\_and\_salaries, P);

int time\_end = clock();

if (print == 1)

print\_vector\_pair(skills\_and\_salaries, P.size());

find(skills\_and\_salaries, Z\_max);

if (print == 1)

{

cout << endl << "Ответ:" << endl;

print\_vector\_pair(skills\_and\_salaries, P.size());

}

return time\_end - time\_start;

}

void find(vector < Skills\_and\_salaries >& skills\_and\_salaries, int Z\_max)

{

int salary\_max = 0;

int skills\_max = 0;

vector < Skills\_and\_salaries > array;

for (int i = 0; i < skills\_and\_salaries.size(); i++)

{

if (skills\_and\_salaries[i].sum\_skills > skills\_max && skills\_and\_salaries[i].sum\_salaries < Z\_max)

{

skills\_max = skills\_and\_salaries[i].sum\_skills;

}

}

for (int i = 0; i < skills\_and\_salaries.size(); i++)

{

if (skills\_and\_salaries[i].sum\_skills == skills\_max)

{

array.push\_back(skills\_and\_salaries[i]);

}

}

skills\_and\_salaries = array;

}

int in\_count\_ckills()

{

cout << "Введите максимальное количество навыков у человека: ";

unsigned short lS\_max = input\_value(rand() % 8 + 4);

cout << "Максимальное количество навыков у человека: " << lS\_max << endl;

return lS\_max;

}

int input\_count\_people()

{

cout << "Введите количество человек: ";

unsigned short N = input\_value(rand() % 8 + 4);

cout << "Количество человек: " << N << endl;

return N;

}

void fill\_pair(vector < pair < set <int>, int> > & P, unsigned short N, unsigned short lS\_max, unsigned short Z\_max)

{

unsigned short lS, M\_max = lS\_max + 2;

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

{

lS = rand() % (lS\_max) + 1;

set <int> S;

for (int j = 0; j != lS; j++)

{

S.insert(rand() % M\_max + 1);

}

P[i] = make\_pair(S, ((Z\_max - 10) / (M\_max)) \* lS + rand() % 10);

}

}

void print\_dash(int count)

{

cout << '|';

for (int j = 0; j < 2; j++)

{

cout << '-';

}

cout << '|';

for (int j = 0; j <= 4 \* (count) - 2; j++)

{

cout << '-';

}

cout << '|';

}

void print\_three\_vector(vector < pair < set <int>, int> > P, unsigned short lS\_max)

{

print\_dash(lS\_max);

cout << endl;

set <int> ::iterator it;

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

{

cout << '|' << setw(2) << i << '|';

for (it = P[i].first.begin(); it != P[i].first.end(); it++)

{

cout << setw(3) << \*it << '|';

}

for (int j = 0; j < lS\_max - P[i].first.size(); j++)

{

cout << setw(4) << '|';

}

//cout << setw(4 \* (N - skills\_and\_salaries[i].first.size() + 3)) << "| зп = " << skills\_and\_salaries[i].second << endl;

cout << " зп =" << setw(4) << P[i].second << endl;

print\_dash(lS\_max);

cout << endl;

}

}

void combinations(int& count, int last, vector <int> indexes, vector < Skills\_and\_salaries >& skills\_and\_salaries, vector < pair < set <int>, int> > P)

{

set <int> unification = {};

int sum\_salaries = 0;

for (int index = 0; index < indexes.size(); index++)

{

copy(P[indexes[index]].first.begin(), P[indexes[index]].first.end(), inserter(unification, unification.begin()));

sum\_salaries += P[indexes[index]].second;

count++;

}

if (indexes.size() != 0)

{

Skills\_and\_salaries sk\_and\_sa;

sk\_and\_sa.index = skills\_and\_salaries.size();

sk\_and\_sa.push(indexes, sum\_salaries, unification.size());

skills\_and\_salaries.push\_back(sk\_and\_sa);

}

for (int index = last + 1; index <= P.size(); index++)

{

indexes.push\_back(index - 1);

combinations(count, index, indexes, skills\_and\_salaries, P);

indexes.pop\_back();

}

}

void print\_vector\_pair(vector < Skills\_and\_salaries > skills\_and\_salaries, int size)

{

if (size > 9)

{

size += (size - 9);

cout << setw(size - 1) << "Элементы:" << '|' << setw(8) << "Индекс:" << '|' << setw(15) << "Сумма навыков:" << '|' << setw(15) << "Сумма зарплат:" << '|' << endl;

for (int i = 0; i < skills\_and\_salaries.size(); i++)

{

int k = 0;

for (int j = 0; j < skills\_and\_salaries[i].indexes.size(); j++)

{

if (skills\_and\_salaries[i].indexes[j] > 9)

k++;

cout << skills\_and\_salaries[i].indexes[j];

}

cout << setw(size - skills\_and\_salaries[i].indexes.size() - k) << '|' << setw(8) << skills\_and\_salaries[i].index << '|' << setw(15) << skills\_and\_salaries[i].sum\_skills << '|' << setw(15) << skills\_and\_salaries[i].sum\_salaries << '|' << endl;

}

}

else

{

cout << setw(9) << "Элементы:" << '|' << setw(8) << "Индекс:" << '|' << setw(15) << "Сумма навыков:" << '|' << setw(15) << "Сумма зарплат:" << '|' << endl;

for (int i = 0; i < skills\_and\_salaries.size(); i++)

{

for (int j = 0; j < skills\_and\_salaries[i].indexes.size(); j++)

{

cout << skills\_and\_salaries[i].indexes[j];

}

cout << setw(size - skills\_and\_salaries[i].indexes.size() + 1 + 9 - size) << '|' << setw(8) << skills\_and\_salaries[i].index << '|' << setw(15) << skills\_and\_salaries[i].sum\_skills << '|' << setw(15) << skills\_and\_salaries[i].sum\_salaries << '|' << endl;

}

}

}

Листинг lab\_4\_2\_functions.h:

#pragma once

#include <iostream>// библиотека

#include <iomanip>

#include <Windows.h> // библиотека

#include "various\_functions.h"

#include "string"

#include "vector"

#include "Skills\_and\_salaries.h"

#include <set>

#include <iterator>

using namespace std; //подключение пространства имен std

int find\_time(unsigned int size, bool print);

void find(vector < Skills\_and\_salaries >& skills\_and\_salaries, int Z\_max);

void combinations(int& count, int last, vector <int> b, vector < Skills\_and\_salaries >& skills\_and\_salaries, vector < pair < set <int>, int> > P);

int in\_count\_ckills();

int input\_count\_people();

void fill\_pair(vector < pair < set <int>, int> >& P, unsigned short N, unsigned short lS\_max, unsigned short Z\_max);

void print\_dash(int count);

void print\_three\_vector(vector < pair < set <int>, int> > P, unsigned short lS\_max);

void print\_vector\_pair(vector < Skills\_and\_salaries > skills\_and\_salaries, int M\_max);

Листинг Skills\_and\_salaries.h:

#pragma once

#include <vector>

#include <Windows.h> // библиотека

using std::vector;

struct Skills\_and\_salaries

{

vector <int> indexes;

int sum\_salaries, sum\_skills, max\_size, index;

void push(vector <int> index\_array, int sum\_sa, int sum\_sk)

{

indexes = index\_array;

sum\_salaries = sum\_sa;

sum\_skills = sum\_sk;

}

};

Листинг main.cpp:

#include <iostream>// библиотека

#include <Windows.h> // библиотека

#include "main\_header.h"

#include "various\_functions.h"

int main()

{

setlocale(0, ""); // поддержка кириллицы в консоли (вывод)

SetConsoleCP(1251); // поддержка кириллицы в консоли (ввод)

SetConsoleOutputCP(1251); // поддержка кириллицы в консоли (ввод)

double task\_number;

boolean debugging = true;

if (debugging == true)

{

//lab\_2\_1();

//lab\_5\_1();

//lab\_2\_2();

lab\_4\_2();

//lab\_4\_1();

//dop\_task\_1();

//lab\_3\_1();

}

else

{

std::cout

<< "Готовые лабораторные работы: " << std::endl

<< std::endl

<< " 1.1 Поиск макс. суммы двух элементов " << std::endl

<< " 1.2 Цикл с вычислением функции " << std::endl

<< " 1.3 Поиск неповторяющихся остатков от деления " << std::endl

<< " 2.1 Количество скобок " << std::endl

<< " 2.2 Бинарный поиск строки по хэшу " << std::endl

<< " 3.1 Рекурсия " << std::endl

<< " 4.1 Оценка алгоритма (Гаусс) " << std::endl

<< " 4.2 База данных " << std::endl

<< " 5.1 Битовая последовательность " << std::endl

<< std::endl

<< "Введите номер лабораторной работы: ";

while (true)

{

task\_number = get\_number<double>();

if (task\_number == 1.1 ||

task\_number == 1.2 ||

task\_number == 1.3 ||

task\_number == 2.1 ||

task\_number == 2.2 ||

task\_number == 3.1 ||

task\_number == 4.1 ||

task\_number == 4.2 ||

task\_number == 5.1)

{

system("cls");

std::cout << "Лабораторная работа № " << task\_number << std::endl;

break;

}

else std::cout << "Такой лабораторной работы нет, введите другой номер: ";

}

if (task\_number == 1.1)

{

lab\_1\_1();

}

else if (task\_number == 1.2)

{

lab\_1\_2();

}

else if (task\_number == 1.3)

{

lab\_1\_3();

}

else if (task\_number == 2.1)

{

lab\_2\_1();

}

else if (task\_number == 2.2)

{

lab\_2\_2();

}

else if (task\_number == 3.1)

{

lab\_3\_1();

}

else if (task\_number == 4.1)

{

lab\_4\_1();

}

else if (task\_number == 4.2)

{

lab\_4\_2();

}

else if (task\_number == 5.1)

{

lab\_5\_1();

}

}

return 0;

}

Листинг various\_functions.h:

#pragma once

#include <string>

using std::string;

using std::cout;

using std::cin;

template <typename T>

T get\_number\_input\_after(string str);

template <typename T>

T input\_value(T a);

template <typename T>

T get\_number();

template <typename T>

bool check(string str);

template <typename T>

bool check\_without\_print(string str);

template <typename T>

T str\_to\_number(string str);

string char\_to\_string(char ch);

template <typename T>

T input\_value(T a)

{

string str;

getline(cin, str);

if (str[0] == '\0')

return a;

return get\_number\_input\_after<T>(str);

}

template <typename T>

T get\_number\_input\_after(string str)

{

while (true) // цикл продолжается до тех пор, пока пользователь не введет корректное значение

{

if (check<T>(str) == 1)

{

return str\_to\_number<T>(str);

}

getline(cin, str);

}

}

template <typename T>

T get\_number()

{

while (true) // цикл продолжается до тех пор, пока пользователь не введет корректное значение

{

string str;

getline(cin, str);

if (check<T>(str) == 1) //проверка на число

{

return str\_to\_number<T>(str); //преобразование строки в число

}

}

}

template <typename T>

bool check(string str)

{

bool k = 1;

if ((str.find('-') < 1 || str.find('-') == -1) && ((str.find('.') < str.size() - 1 && str.find('.') > 0) || str.find('.') == -1))

{

if (str.length() > 1)

{

int count\_minus = 0, count\_point = 0;

for (int i = 0; i < str.length(); i++)

{

if (str[i] == '-')

{

count\_minus++;

}

if (str[i] == '.')

{

count\_point++;

}

}

if ((str[0] == '0' && str[1] != '.') || count\_point > 1 ||

(str[0] == '-' && str[1] == '.') ||

(str[0] == '-' && str[1] == '0' && str[2] != '.') || count\_minus > 1)

{

k \*= 0;

std::cout << "!!Вы ввели не число!!" << std::endl << "Введитe число: ";

}

}

else

if (str == "-")

{

k \*= 0;

std::cout << "!!Вы ввели не число!!" << std::endl << "Введитe число: ";

}

if (k == 1)

{

string s = typeid(T).name();

if (s == "double")

{

for (int i = 0; i < str.length(); i++)

{

if ((str[i] < '0' || str[i] > '9') && str[i] != '.' && str[i] != '-')

{

k \*= 0;

std::cout << "!!Вы ввели не число!!" << std::endl << "Введитe число: ";

break;

}

}

}

if (s == "int")

{

for (int i = 0; i < str.length(); i++)

{

if ((str[i] < '0' || str[i] > '9') && str[i] != '-')

{

k \*= 0;

std::cout << "!!Вы ввели не целое число!!" << std::endl << "Введитe целое число: ";

break;

}

}

}

if (s == "bool")

{

if (str[0] != '0' && str[0] != '1')

{

k \*= 0;

std::cout << "!!Вы ввели не двоичное число!!" << std::endl << "Введитe двоичное число: ";

}

}

}

}

else

{

k \*= 0;

std::cout << "!!Вы ввели не число!!" << std::endl << "Введитe число: ";

}

return k;

}

template <typename T>

bool check\_without\_print(string str)

{

bool k = 1;

// 67642.5423

//проверка на возможность существования такого числа

if ((str.find('-') < 1 || str.find('-') == -1) && ((str.find('.') < str.size() - 1 && str.find('.') > 0) || str.find('.') == -1))

{

//если длина больше

if (str.length() > 1)

{

//проверка на возможность существования такого числа

//после нуля всегда точка или ничего, если ноль первый

//после - если ноль, то должна быть точка или ничего

int count\_minus = 0, count\_point = 0;

for (int i = 0; i < str.length(); i++)

{

if (str[i] == '-')

{

count\_minus++;

}

if (str[i] == '.')

{

count\_point++;

}

}

if ((str[0] == '0' && str[1] != '.') || count\_point > 1 ||

(str[0] == '-' && str[1] == '.') ||

(str[0] == '-' && str[1] == '0' && str[2] != '.') || count\_minus > 1)

{

k \*= 0;

std::cout << "!!Вы ввели не число!!" << std::endl << "Введитe число: ";

}

}

else

if (str == "-")

{

k \*= 0;

}

if (k == 1)

{

string s = typeid(T).name();

if (s == "double")

{

for (int i = 0; i < str.length(); i++)

{

if ((str[i] < '0' || str[i] > '9') && str[i] != '.' && str[i] != '-')

{

k \*= 0;

break;

}

}

}

if (s == "int")

{

for (int i = 0; i < str.length(); i++)

{

if ((str[i] < '0' || str[i] > '9') && str[i] != '-')

{

k \*= 0;

break;

}

}

}

}

}

else

{

k \*= 0;

}

return k;

}

template <typename T>

T str\_to\_number(std::string str)

{

T a = 0; //само число

int index = 0, k = 1;

if (str[0] == '-')

{

k = -1;

str.erase(0, 1);

}

if (str.find('.') != -1)

{

index = str.find('.');

str.erase(str.find('.'), 1);

}

else

index = str.length();

for (int i = 0; i < str.length(); i++)

{

a = a \* 10 + (str[i] - '0');

}

return k \* (a / pow(10, (str.length() - index)));

}

Листинг various\_functions.cpp:

#include <iostream>// библиотека

#include <Windows.h> // библиотека

#include "various\_functions.h"

#include <vector>

using namespace std;

string char\_to\_string(char ch)

{

string s = {};

s += ch;

return s;

}

Листинг various\_functions.cpp:

#pragma once

void dop\_task\_1();

void lab\_1\_1();

void lab\_1\_2();

void lab\_1\_3();

void lab\_2\_1();

void lab\_2\_2();

void lab\_3\_1();

void lab\_4\_1();

void lab\_4\_2();

void lab\_5\_1();

Результат.

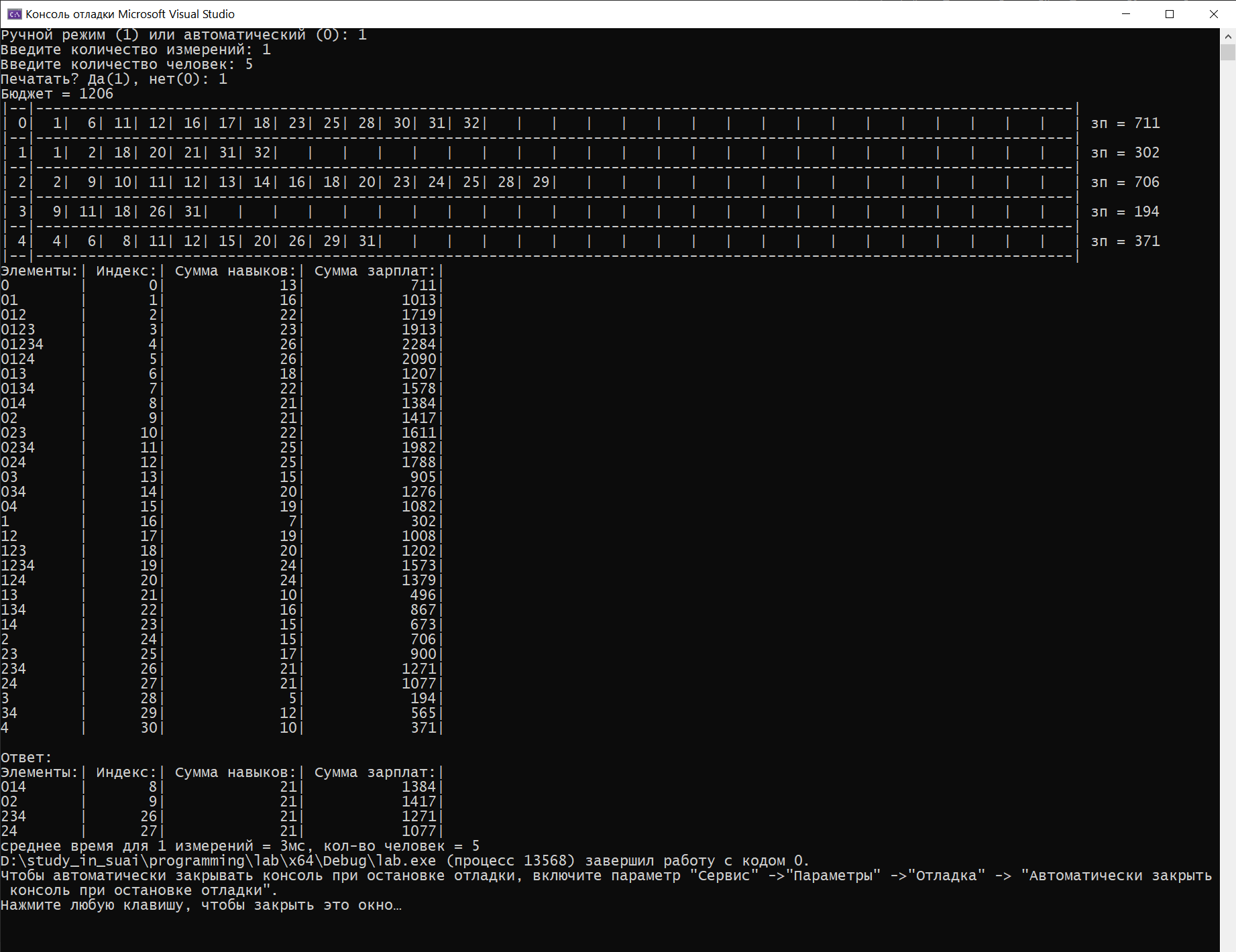
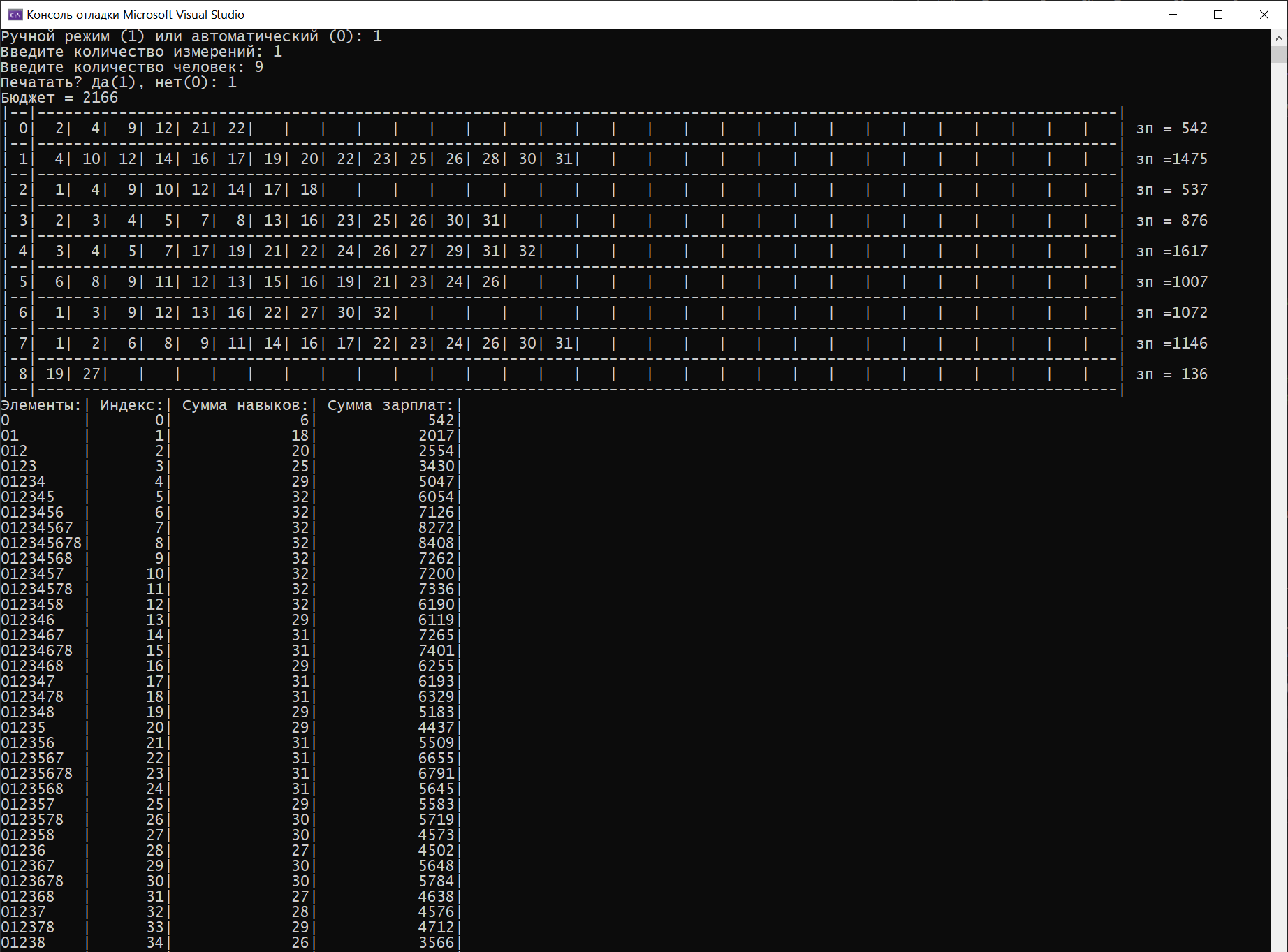


Рисунок 2. Тест



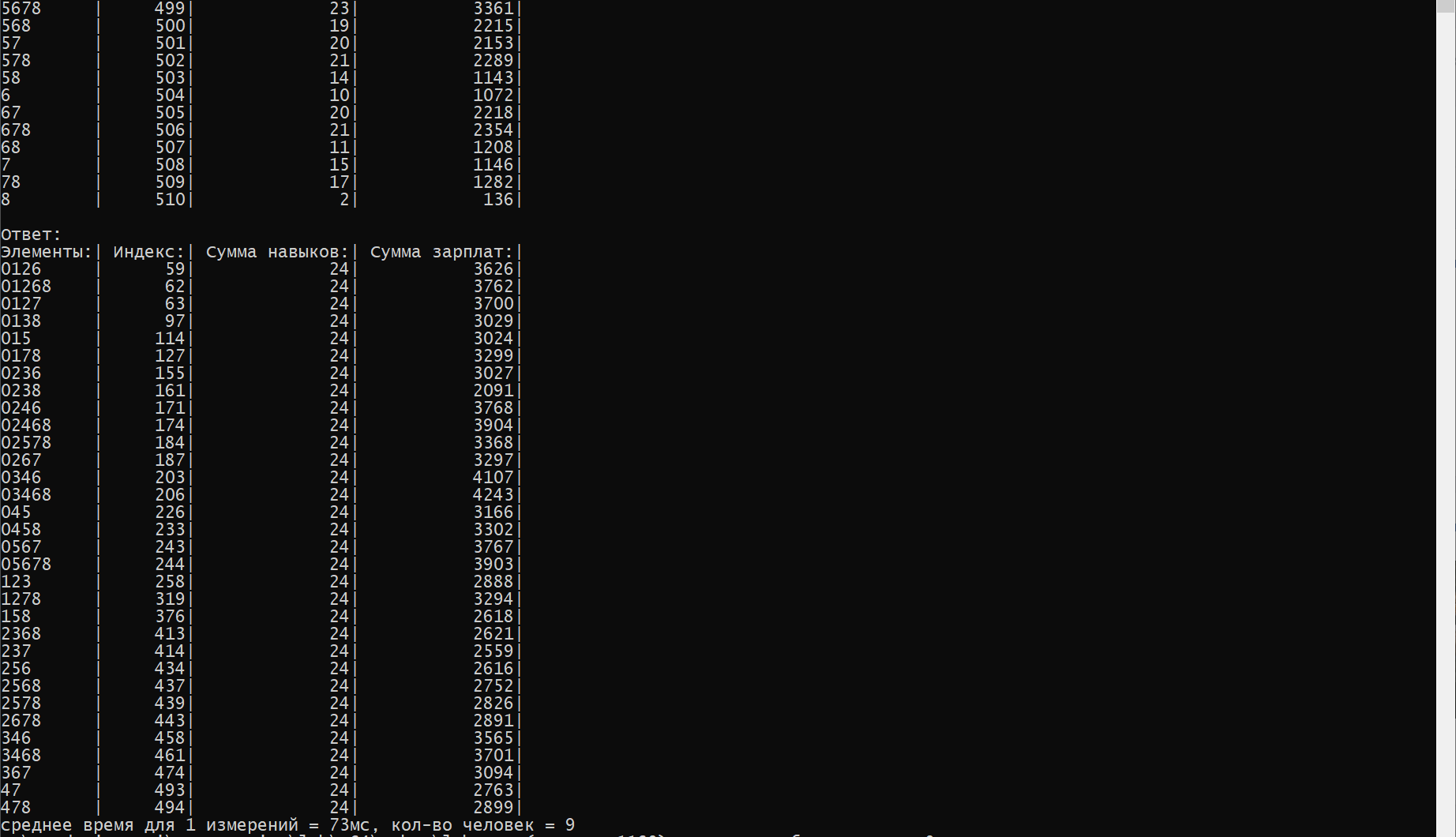
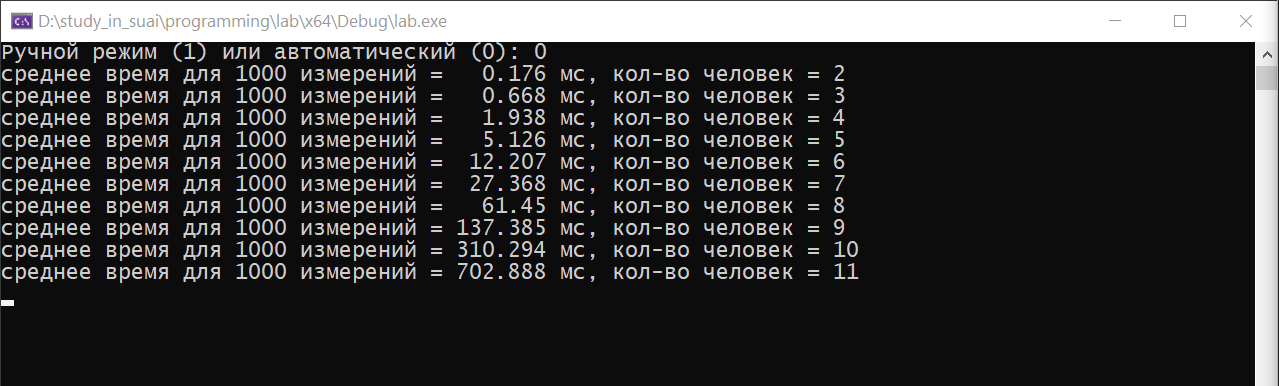


Рисунок 3. Тест

Рисунок 4. Тест

Оценка времени и сложности

Временная сложность:



Оценка сложности:

Сложность O(2^n). Она получается от зависимости размера полученного массива от времени. При количестве человек 5, длина массива получается 31.

# Вывод.

Вводится массив пар навыков персонажей и их зарплат. Потом находится массив всех возможных сумм навыков и зарплат персонажей. Потом находится максимально возможая сумма зарплат, не превышающая Z\_max, когда сумма навыков тоже максимальная. Потом в выходной массив вводятся все персонажи, сумма навыков которых, равна максимальной.