ПРИЛОЖЕНИЕ № 3
к Порядку организации
практической подготовки обучающихся,
утверждённому приказом от
«____»_____20___г. № _____

Федеральное государственное автономное образовательное
учреждение высшего образования
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт математики, механики и компьютерных наук им И.И. Воровича
структурное подразделение

01.03.02 Прикладная математика и информатика
специальность/направление подготовки (код, наименование)

ОТЧЁТ
о прохождении практики

обучающегося 3 курса

Фамилия _____

Имя _____

Отчество (при наличии) _____

Место практики Институт математики, механики и компьютерных наук им И.И. Воровича
наименование профильной организации /структурного подразделения Университета

Вид практики: учебная
учебная/производственная

Тип практики: учебная
указывается в соответствии с ОПОП

Способ проведения практики: стационарная
стационарная/выездная

Сроки прохождения практики: *с 01.07.2021 по 14.07.2021*

Задание обучающегося на практику согласовано*:

Руководитель практики
от Университета
Баглий Антон Павлович
_____
подпись, Ф.И.О.

Руководитель практики
от профильной организации

подпись, Ф.И.О.

I. ЗАДАНИЕ ОБУЧАЮЩЕГОСЯ НА ПРАКТИКУ

1. Изучить алгоритмы решения вычислительно трудных задач
2. Разработать алгоритм, применяющий метод рекурсии к задаче о ребусах.
3. Разработать программу для решения ребусов о сложении слов.

II. ИНСТРУКТАЖ ПО ОЗНАКОМЛЕНИЮ С ТРЕБОВАНИЯМИ ОХРАНЫ ТРУДА, ТЕХНИКИ БЕЗОПАСНОСТИ, ПОЖАРНОЙ БЕЗОПАСНОСТИ, ПРАВИЛАМ ВНУТРЕННЕГО РАСПОРЯДКА

	Инструктаж проведен	**Ознакомлен**
по требованиям охраны труда по технике безопасности по пожарной безопасности по правилам внутреннего трудового распорядка	_____ *(подпись и Ф.И.О руководителя практики от Университета,)* «01» июля 2021г.	_____ *(подпись и Ф.И.О. обучающегося)* «____» _____ 20__г.

III. ДНЕВНИК ПРАКТИКИ

Дата	Выполненные мероприятия в соответствии с заданием на практику
01.07.21	Прослушивание инструктажа по технике безопасности, заполнение журнала по технике безопасности.
02.07.21	Прослушивание вводных лекций. Получение индивидуальных заданий. Консультации по заданиям.
06.07.21	Реализация рекурсивного варианта программы
10.07.21	Реализация варианта программы на основе очереди заданий
13.07.21	Заполнение дневника и отчета об учебной практике. Защита работы.

IV. АНАЛИЗ ПРОВЕДЁННОЙ РАБОТЫ В ПЕРИОД ПРОХОЖДЕНИЯ ПРАКТИКИ ОБУЧАЮЩИМСЯ

Раздел заполняется обучающимся в соответствии со спецификой практики (может содержать таблицы, графики, статистические данные и т.п.)

№ п/п	Выполненные мероприятия в соответствии с заданием на практику	Анализ проведенной работы
1	Изучение алгоритмов для решения полученной задачи. Разработка сервисных процедур и функций, структур данных.	Разработаны сервисные процедуры/функции и (загрузка данных, вывод результатов, внутреннее представление для обрабатываемых данных)
2	Реализация рекурсивного варианта программы	Реализован вариант программы на основе рекурсии, решающий поставленную задачу

Постановки задачи

Написать программу решения математических ребусов, в которых зашифровано сложение. Написать программу, которая получает на входе математический ребус в виде трех слов и возвращает числа, которые были представлены этими словами, или сообщает о невозможности решения.

– Используя рекурсию.
Необходимо самостоятельно подготовить тестовые входные данные.

Требования к проекту:
– Язык: C, C++
– Среда разработки: Visual Studio, Code Blocks, Qt Creator, PyCharm.
– Версии должны быть совместимы с теми, что установлены в компьютерных классах.
– Приложение должно работать через командную строку.
– Выходные и выходные данные — в текстовых файлах

Общие понятия

Числовой ребус (математический ребус) – также арифметический ребус, крипторитм (cryptarithm), альфаметик (alphametic) — математическая головоломка, пример арифметического действия, в котором все или некоторые цифры заменены буквами, звёздами или другими символами. Задание состоит в том, чтобы восстановить исходную запись примера. Числовые ребусы бывают нескольких видов, например:
– Цифры в записи вычисления заменены буквами; одинаковые буквы соответствуют одинаковым цифрам. Буквы могут образовывать существующие слова.
– Некоторые цифры в записи заменены «звездочками».
– В одном примере могут использоваться и буквы, и «звёздочки».
Задача сводится к восстановлению полной записи вычислений. Некоторые числовые ребусы имеют несколько вариантов решения. При разгадывании числовых ребусов обычно условием ставится проверка всех возможных вариантов.

Классический пример, опубликованный в июле 1924 года в журнале Strand Magazine: [1]

```
    S E N D
+   M O R E
-----------
= M O N E Y
```

Методы решения

Программа работает методом перебора вариантов. Изначально в нее поступает 3 слова (из файла): первое слагаемое, второе слагаемое и сумма.

Сначала, происходить проверка длин трех исходных слов на корректность. Дальше, мы составляем «алфавит»: вектор, состоящий из всех различных букв данных слов. Потом, из всех использованных букв создаётся словарь (map), где каждой букве сопоставляется одна цифра. Далее, используя этот словарь, мы каждое исходное полученное слово превращаем в число и делаем проверку: выполняется ли равенство между суммой этих двух полученных слагаемых и числом, полученным из третьего исходного слово, преобразованного, используя наш словарь.

Если равенство выполняется (и каждой отдельной букве соответствуют разные цифры, первая цифра в любом числе может быть нулем), то значить мы нашли подходящую комбинацию цифр. Но мы не завершаем работу программы, а продолжаем и дальше искать другие решения до тех пор, пока мы не получим и не проверим все возможные комбинации цифр соответствующим разным буквам.

В конце работы программы, найденные решения будут печататься в новый файл, а если не нашлись решения – печатается соответствующее сообщение.

Программная реализация

В программе используется 8 функций (включая main).

Функция decode используется для составления алфавита и последующего вызова основной перебирающей функции. На вход ей подаются: слова, взятые из файла (a, b, c). Если длина суммы окажется меньше первой или второй слагаемого, то программа выводит сообщение о невозможности нахождения ответов и завершается, иначе продолжит работу. Сначала мы объединяем все слова в одну строку, потом из этой полученной строки создаём вектор уникальных букв. Далее, вызывается функция find.

Функция find отвечает за перебор значений и сопоставление их элементам каждой полученного слова. На вход она принимает: изначально данные слова и словарь букв и цифр (map). Работа функции find заключается в том, чтобы проверить полученную комбинацию и сгенерировать неповторяющиеся комбинации различных цифр и отправить их снова себе (изначально, получает на вход словарь, где каждой букве соответствует цифра 0). Если

какая-то комбинация подошла, то эта комбинация печатается в файл (output), иначе продолжаем генерации новой комбинации и следя за тем, чтобы значения не выходили за определенный предел. Если все возможные комбинации проверены, то функция заканчивает работу.

Функция проверки print_numbers печатает полученные на вход три строки и печатает их в файл.

Функция same_val проверят, что всем ли буквам в словаре соответствуют разные цифры и возвращает булево значение.

Функция MyAtoi переводит полученную string (состоящую из цифр) в int. Например: string s = "052" → int s = 52.

Функция decode_string, в соответствие с полученным на входе словарём cipher, из строки s (тоже полученную на входе, из буквенного вида), переводить в цифровой. Например, s = ABC; cipher = {"A" - 0, "B" - 5, "C" - 2}. Тогда, результатом работы этой функции будет: string s = "052".

Функция is_declared, используется при составлении множество всех использованных букв в трех исходных слов. Проверяет, существует ли в слове v, символ i.

Для измерения времени работы программы использовалось

clock_t start_c = clock() - в начале программы и

clock_t end_c = clock() - в конце

(double) (end_c - start_c) / CLOCKS_PER_SEC - так и узнавалось время работы.

В результате работы программы на входных данных мы получаем текстовый файл, в строках которого распечатаны исходная задача, всевозможные решения, текстовое сообщение и время работы программы.

Код:

```
#include <iostream>
#include <map>
#include <string>
#include <vector>
#include <algorithm>
#include <iterator>
#include <fstream>
#include <ctime>
#include <cmath>

using namespace std;
string glob_a, glob_b, glob_c;
bool flag;
map <char, int> glob_cipher;

/* проверка, существует ли в слове v, символ i */
bool is_declared(char& i, vector<char>& v) {
	bool dec = false;
	for (char j : v) {
		dec = dec || (i == j);
	}
	return dec;
}
/* в соответствие со словарём cipher, из строки s (из буквенного вида),
   переводить в цифровой.
   Например, s = ABC; cipher = {"A" - 0, "B" - 5, "C" - 2};
   Тогда, результатом работы этой функции будет:
   string s = "052" */
void decode_string(string& s, map <char, int>& cipher) {
	map <char, int> ::iterator it = cipher.begin();
	for (int i = 0; i < (int)s.size(); i++) {
		it = cipher.find(s[i]);
		s[i] = it->second + '0';
	}
}
/* эта функция, переводит из string в int полученную на вход строку:
   Например: string s = "052" --> int s = 52 */
int MyAtoi(string& a) {
	int s = 0, d = 0;
	for (int i = a.size() - 1; i >= 0; i--) {
		char tmp = a[i];
		s += (int)((int)tmp - 48) * (int)pow(10, d);
		d++;
	}
	return s;
}
/* эта функция проверяет, всем ли буквам сопоставлены разные цифры */
bool same_val(map <char, int>& cipher) {
	map <char, int> ::iterator it = cipher.begin();
	for (int i = 0; it != cipher.end(); it++, i++) {
		map <char, int> ::iterator it1 = cipher.begin();
		for (int j = 0; it1 != cipher.end(); it1++, j++) {
			if (it->second == it1->second && it1->first != it->first) return
false;
		}
	}
	return true;
}
/* печать полученных на вход строк в файл */
void printNumbers(string& a, string& b, string& c) {
	ofstream fout("output.txt", ios_base::app);
	fout << a << " + " << b << " = " << c << endl;
	fout.close();
}
```

```
/* вторая расшифровывающая функция, которая делает перебор всех
   возможных вариантов - рекурсивная */
void find(string& a, string& b, string& c, map <char, int>& cipher) {
	/*согласно словарю, буквенные слова конвертируем в цифровую строку */
	decode_string(a, cipher);

	decode_string(b, cipher);
	decode_string(c, cipher);
	/* цифровые строки переводим в число */
	int aInt = MyAtoi(a), bInt = MyAtoi(b); int cInt = MyAtoi(c);
	/*if (aInt < pow(10, glob_a.size() - 1))
		return;
	if (bInt < pow(10, glob_b.size() - 1))
		return;
	if (cInt < pow(10, glob_c.size() - 1))
		return;*/
	int expression = aInt + bInt;
	/* проверка результата сложения и проверка, всем ли буквам в словаре
	   сопоставлены разные цифры */
	if (expression == cInt && same_val(cipher)) {
		a = to_string(aInt), b = to_string(bInt), c = to_string(cInt);
		/* если да, то печать результата в файл */
		printNumbers(a, b, c);
		/* ставим, флаг если нашли решение, это поможет, чтобы в конце работы
		   программы печатать соответствующее сообщение */
		flag = true;
	}
	/* получаем следующую комбинацию цифр, соответствующих буквам словаря */
	int sum = 0;
	map <char, int> ::iterator it = cipher.begin();
	for (int i = 0; it != cipher.end(); it++, i++) {
		sum += it->second * (int)pow(10, cipher.size() - 1 - i);
	}
	sum += 1;
	/* если, все возможные комбинации цифр соответствующим длинам
	   входных слов перебраны, то заканчиваем работу программу с
	   соответствующим сообщением */
	if (sum >= pow(10, cipher.size())) {
		if (flag)
			throw runtime_error("\nThere are no other solutions");
		throw runtime_error("There are no solution");
	}
	/* новую полученную комбинацию цифр записываем в словарь */
	int p = (int)pow(10, cipher.size() - 1);
	for (map <char, int> ::iterator it1 = cipher.begin(); it1 != cipher.end();
it1++) {
		it1->second = sum / p;
		sum -= it1->second * p;
		p /= 10;
	}

	/* вернём исходные значения переменных обратно,
	   т.к. были внесены изменения */
	a = glob_a;
	b = glob_b;
	c = glob_c;
	/* рекурсивно вызываем эту функцию */
	find(a, b, c, cipher);
}
```

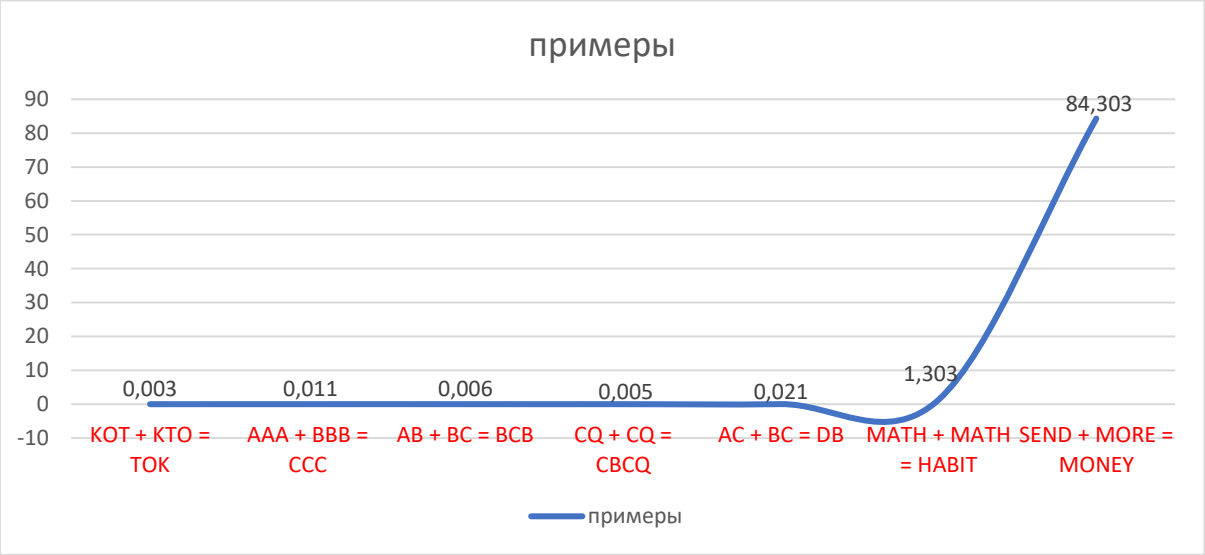
```
/* первая расшифровывающая функция */
void decode(string& a, string& b, string& c) {
	/* проверка: длины входных слов */
	if ((c.size() < b.size()) or (c.size() < a.size())) throw runtime_error("Нет
решения");
	string d = a + b + c;
	vector <char> carr;
	for (int i = 0; i < d.size(); i++) {
		if (!is_declared(d[i], carr)) carr.push_back(d[i]);
	}
	/* получили carr - множество использованных букв */
	/* далее, сордаём map - сопоставление каждой букве одну цифру */
	map <char, int> cipher;
	for (int i = 0; i < carr.size(); i++) {
		cipher.insert(make_pair(carr[i], 0));
	}
	glob_cipher = cipher;
	/* если, нижестоящий цикл оставить, то получим не рекурсивный, а итеративный
	   метод решения */
	/*while (true) {
		find(a, b, c, cipher);
		a = glob_a;
		b = glob_b;
		c = glob_c;
	}*/
	/* вызываем вторую расшифровывающую функцию */
	find(a, b, c, cipher);
}

int main() {
	setlocale(LC_ALL, "Russian");
	clock_t start_c, end_c;
	start_c = clock();
	try {
		/* считываем входные данные из файла */
		char file[50] = "input7.txt";
		string a, b, c;
		ifstream fin(file);
		fin >> a;
		fin >> b;
		fin >> c;
		fin.close();
		/* сохраняем в глобальные переменные входные данные,
		   чтобы после восстановить значения, после изменений */
		glob_a = a;
		glob_b = b;
		glob_c = c;
		/* в выходной файл записываем само условие задачи и вызываем
		   расшифровывающую функцию */
		ofstream fout("output.txt");
		fout << a << " + " << b << " = " << c << "\n" << endl;
		fout.close();
		decode(a, b, c);
	}
	catch (exception& e) {
		/* при выявление исключений, или окончания расшифровки - печатаем в файл
		   сообщение */
		ofstream fout("output.txt", ios_base::app);
		fout << e.what() << endl;
		fout.close();
	}
	end_c = clock();
```

```
        /* печатаем время работы программы */
        ofstream fout("output.txt", ios_base::app);
        fout << "runtime = " << (double)(end_c - start_c) / CLOCKS_PER_SEC << endl;
        fout.close();
        return 0;
}
```

Результаты

График зависимости времени от количества переменных и числа используемых потоков.



Протестировав программу для разных входных данных, был построен график, который показывает время работы программы при разном количестве переменных.

Также для проверки правильности решения были использованы входные данные небольшой величины, для простого подсчета верного ответа.

```
CQ + CQ = CBCQ

There are no solution
runtime = 0.005
```

```
AB + BC = BCB

91 + 10 = 101

There are no other solutions
runtime = 0.006
```

```
MATH + MATH = HABIT

7521 + 7521 = 15042

There are no other solutions
runtime = 1.303
```

```
KOT + KTO = TOK

459 + 495 = 954

There are no other solutions
runtime = 0.003
```

```
SEND + MORE = MONEY

7531 + 825 = 8356
5731 + 647 = 6378
3821 + 468 = 4289
6851 + 738 = 7589
8432 + 914 = 9346
8542 + 915 = 9457
3712 + 467 = 4179
5732 + 647 = 6379
7643 + 826 = 8469
6853 + 728 = 7581
8324 + 913 = 9237
6524 + 735 = 7259
7534 + 825 = 8359
6415 + 734 = 7149
7316 + 823 = 8139
9567 + 1085 = 10652
2817 + 368 = 3185
6419 + 724 = 7143
7429 + 814 = 8243
7539 + 815 = 8354
7649 + 816 = 8465
3719 + 457 = 4176
2819 + 368 = 3187
3829 + 458 = 4287
5849 + 638 = 6487

There are no other solutions
runtime = 84.303
```

```
AAA + BBB = CCC

111 + 222 = 333
111 + 333 = 444
111 + 444 = 555
111 + 555 = 666
111 + 666 = 777
111 + 777 = 888
111 + 888 = 999
222 + 111 = 333
222 + 333 = 555
222 + 444 = 666
222 + 555 = 777
222 + 666 = 888
222 + 777 = 999
333 + 111 = 444
333 + 222 = 555
333 + 444 = 777
333 + 555 = 888
333 + 666 = 999
444 + 111 = 555
444 + 222 = 666
444 + 333 = 777
444 + 555 = 999
555 + 111 = 666
555 + 222 = 777
555 + 333 = 888
555 + 444 = 999
666 + 111 = 777
666 + 222 = 888
666 + 333 = 999
777 + 111 = 888
777 + 222 = 999
888 + 111 = 999

There are no other solutions
runtime = 0.011
```

```
AC + BC = DB

6 + 26 = 32
7 + 47 = 54
8 + 68 = 76
15 + 5 = 20
16 + 26 = 42
12 + 42 = 54
17 + 47 = 64
13 + 63 = 76
14 + 84 = 98
25 + 5 = 30
23 + 63 = 86
28 + 68 = 96
35 + 5 = 40
31 + 21 = 52
32 + 42 = 74
37 + 47 = 84
41 + 21 = 62
46 + 26 = 72
51 + 21 = 72
56 + 26 = 82
52 + 42 = 94
65 + 5 = 70
61 + 21 = 82
75 + 5 = 80
71 + 21 = 92
85 + 5 = 90

There are no other solutions
runtime = 0.021
```

Тесты проводились на ПК с процессором AMD Ryzen 7 PRO 1700X Eight-Core Processor 3.40 GHz, 16ГБ ОЗУ и ОС Windows 10. Программа компилировалась компилятором Microsoft Visual C++ в режиме «release» с параметрами оптимизации по умолчанию.

Список использованных источников

1. Числовой ребус – https://www.wikiwand.com/ru/%D0%A7%D0%B8%D1%81%D0%BB%D0%BE%D0%B2%D0%BE%D0%B9_%D1%80%D0%B5%D0%B1%D1%83%D1%81

ОТЗЫВ РУКОВОДИТЕЛЯ ПРАКТИКИ ОТ ПРОФИЛЬНОЙ ОРГАНИЗАЦИИ *

Отзыв оформляется руководителем практики от профильной организации в свободной форме с указанием полноты, своевременности и качества проведенной обучающимся работы

Руководитель практики
от профильной организации _____ /Баглий Антон Павлович
подпись Ф.И.О.

ОТЗЫВ РУКОВОДИТЕЛЯ ПРАКТИКИ ОТ УНИВЕРСИТЕТА

Отзыв оформляется руководителем практики от Университета в свободной форме с указанием полноты, своевременности и качества проведенной обучающимся работы

Оценка _____
зачтено/отлично/хорошо/удовлетворительно

Руководитель практики
от Университета _____ /Баглий Антон Павлович
подпись Ф.И.О.

Примечания:

1. Отчёт о прохождении практики является основным рабочим и отчётным документом обучающегося в период прохождения практики.
2. Обучающийся заполняет отчёт о прохождении практики регулярно в течение всего периода практики.
3. Заполненный отчёт о прохождении практики обучающийся сдает руководителю практики от Университета по завершению практики в соответствии с графиком учебного процесса.
4. Отчёты о прохождении практики обучающихся хранятся на соответствующей кафедре в течение всего периода реализации образовательной программы.

---

*Заполняется в случае проведения практики в профильной организации