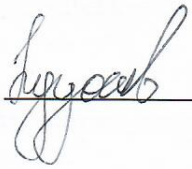


МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Вятский государственный университет»
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

Отчет
по учебной практике

Дата начала прохождения практики: 06.07.2020

Дата окончания прохождения практики: 19.07.2020

Выполнил студент группы ИВТб-1301-01-00  /Я.Ю. Кудяшев/

Киров
2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Вятский государственный университет»
(ВятГУ)

РАБОЧИЙ ГРАФИК (ПЛАН) ПРОВЕДЕНИЯ ПРАКТИКИ

Ф.И.О. обучающегося	Кудяшев Ярослав Юрьевич		
Институт/факультет	Факультет автоматики и вычислительной техники		
Направление подготовки (специальность)	09.03.01 Информатика и вычислительная техника		
Направленность (профиль)	01 Программное и аппаратное обеспечение вычислительной техники		
Вид практики	Учебная практика		
Сроки прохождения практики с	06.07.2020	по	19.07.2020
Место прохождения практики	ФГБОУ ВО «Вятский государственный университет»		

Руководитель практики от университета Чистяков Геннадий Андреевич
(Ф.И.О. руководителя)

Индивидуальные задания, выполняемые в период практики: 1. Разработка графического приложения "Пятнашки".

в период практики:		
Период	Наименование разделов практики и их содержание	Компетенции ¹
Раздел 1 «Подготовительный этап практики»		
06.07.2020	Прохождение инструктажа по охране труда, пожарной безопасности	
	Ознакомление с правилами внутреннего трудового распорядка	
Раздел 2 «Основной этап практики»		ОПК-3, ОПК-8, УК-6
07.07.2020- 16.07.2020	Решение научно-исследовательских задач	
07.07.2020- 16.07.2020	Выполнение индивидуального задания	
Раздел 3 «Заключительный этап практики»		ОПК-3, ОПК-8, УК-6
17.07.2020	Подготовка и оформление отчета по практике	
Раздел «Подготовка и прохождение промежуточной аттестации»		
18.07.2020	Подготовка к защите и защита отчета по практике	

Рабочий график (план) проведения практики сформирован на основе программы практики

ПП 3-09.03.01.04 2019 106181
(номер регистрации)

Руководитель практики от университета

04.07.2020
(дата)


(подпись)

Г.А. Чистяков
(Ф.И.О.)

¹ Компетенции в соответствии с ФГОС ВО:

УК-6: Способен управлять своим временем, выстраивать и реализовывать траекторию саморазвития на основе принципов образования в течение всей жизни

ОПК-3: Способен решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности

ОПК-8: Способен разрабатывать алгоритмы и программы, пригодные для практического применения

Содержание

Введение	3
1 Общая часть	4
1.1 Первая задача	4
1.2 Вторая задача	6
1.3 Третья задача	7
1.4 Четвертая задача	9
1.5 Выводы по общей части	10
2 Индивидуальная часть	11
2.1 Формулировка решаемой задачи	11
2.2 Подходы к решению и результаты	11
2.3 Выводы по индивидуальной части	13
Заключение	14

Введение

Данный документ представляет собой отчет о прохождении учебной практики, предусмотренной образовательной программой «Программное и аппаратное обеспечение вычислительной техники», реализуемой в ФГБОУ ВО «Вятский государственный университет».

Место прохождения практики – ФГБОУ ВО «Вятский государственный университет». Сроки прохождения практики – с 06.07.2020 по 19.07.2020.

Практика включала в себя две части: общую и выполняемую в рамках индивидуального задания.

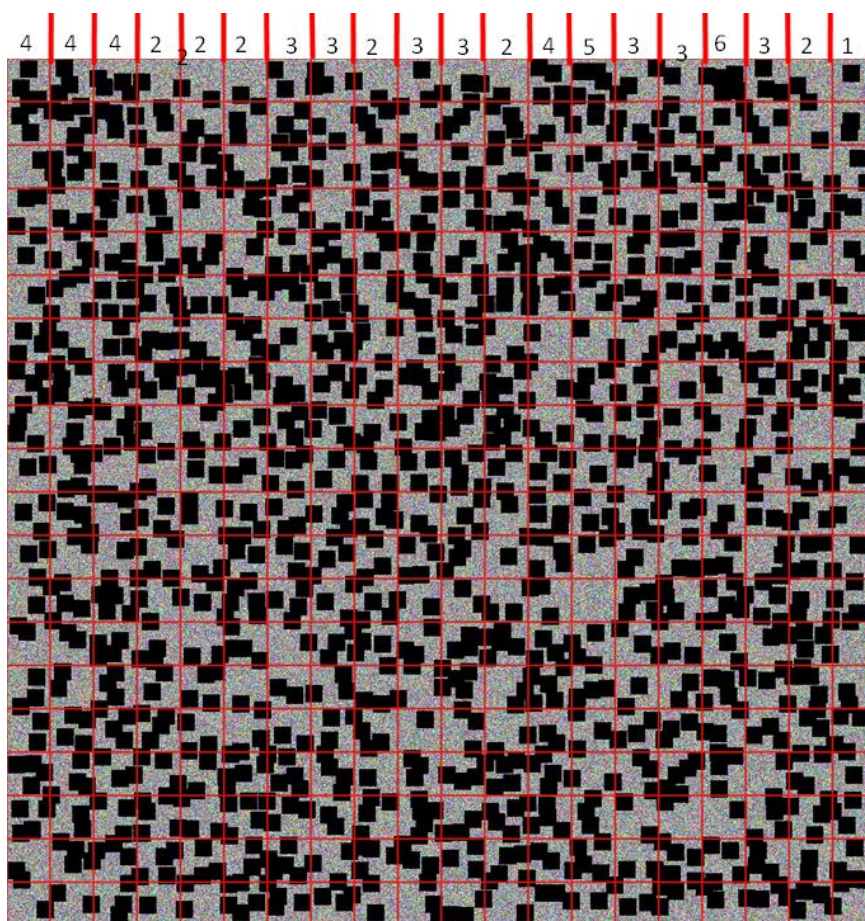
1 Общая часть

В данном разделе рассматриваются вопросы, связанные с прохождением общей для всех обучающихся части практики.

1.1 Первая задача

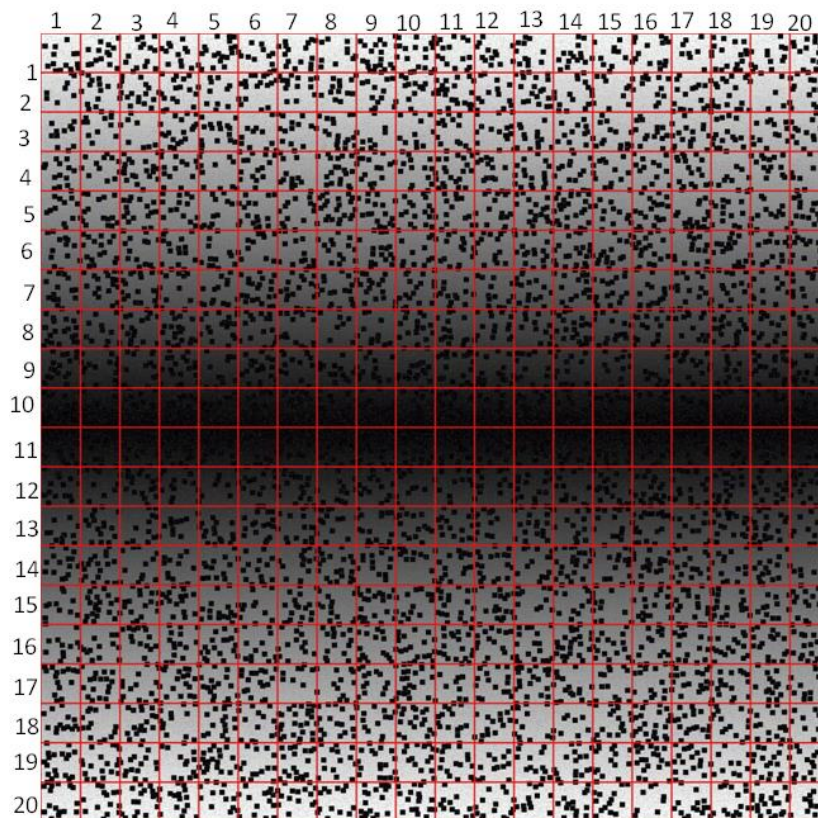
Первое задание в первой задаче было решено по-обычному: подсчитано количество чёрных квадратов. В данном случае данный метод не является особо трудоёмким и времязатратным, т.к. количество квадратов не так велико, как в следующих заданиях. В результате подсчёта количество чёрных квадратов получилось равным 119.

Второе задание в первой задаче уже сложнее и методом простого подсчёта квадратов его придётся решать очень долго. В качестве решения данного задания было решено использовать метод напоминающий метод Монте-Карло. Сначала весь большой квадрат был разделён при помощи сетки с размерностью 20x20. В итоге получился большой квадрат, разделённый на 400 равных секций:



После проделанных действий было подсчитано количество черных квадратов во всех секциях первой строки. Результаты представлены на рисунке. Неполные квадраты, которые входят больше, чем наполовину, были засчитаны за полные квадраты. Этот метод схож с методом Монте-Карло тем, что сначала мы разбиваем изображение при помощи сетки, а затем считаем количество точек в каждой из них. Но это было сделано лишь для первой строки. Затем мы подсчитываем сумму всех квадратов первой строчки и умножаем её на 20, где 20 – размерность сетки. В результате получаем $61 * 20 = 1220$. Понятно, что это лишь приближённое значение, которое может отличаться от истинного.

Третье задание в первой задаче можно также решить методом, который был использован для второго задания, но было решено использовать стохастический метод. Квадрат вновь был разбит на равные секции при помощи сетки 20x20. Затем случайным образом были выбраны 20 секций этого квадрата, в которых было посчитано количество чёрных квадратов.

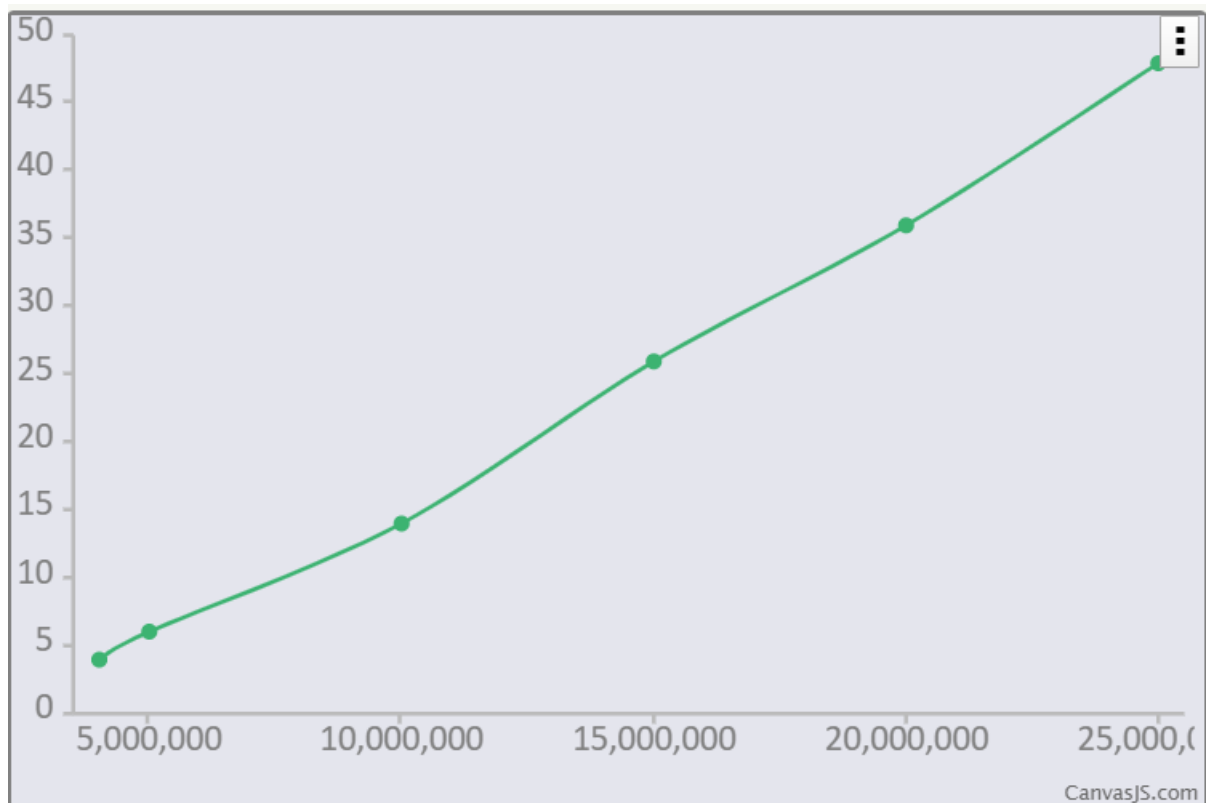


$(1,1) = 13$, $(8,3) = 15$, $(3,19) = 21$, $(8,17) = 17$, $(12,18) = 9$, $(7,2) = 17$,
 $(19,16) = 16$, $(18,18) = 9$, $(2,1) = 22$, $(6,19) = 13$, $(8,20) = 14$, $(3,20) = 18$,
 $(9,1) = 15$, $(15,18) = 11$, $(20,20) = 14$, $(14,14) = 14$, $(18,3) = 18$, $(18,12) = 9$,
 $(7,19) = 17$, $(3,4) = 17$.

Затем была получена сумма данных квадратов и умножена на 20, где 20 – размерность сетки. В итоге получилось приближённое число чёрных квадратов: $299 * 20 = 5980$.

1.2 Вторая задача

Для определения средней асимптотической оценки временной сложности реализованного в программе алгоритма было решено первоначально измерить время, за которое программа выполняет алгоритм. В качестве аргументов, используемых программой, были выбраны значения: 5000000, 10000000, 15000000 и т.д. С данными значениями получился график:



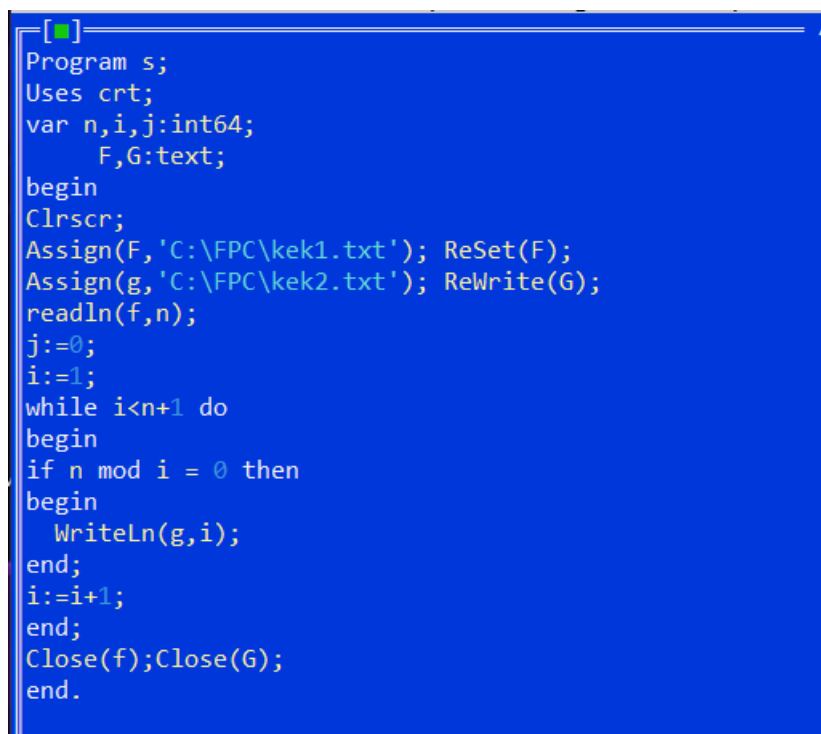
Для меньших значений график практически не изменяется, т.к. время их выполнения практически всегда равно 0,03 сек. На оси ординат время, за которое выполняется определённый алгоритм. Будет логично предположить, что с увеличением времени выполнения алгоритма увеличивается и его сложность. Исходя из данного утверждения и приведённого графика становится ясно, что сложность алгоритма $T(n)$.

Для второго и третьего задания можно использовать похожий подход, но график будет отличаться своей нелинейностью. Во время измерения времени, которое требуется для обработки алгоритмов второго и третьего задания, значения были неравномерными, т.е. вычисления даже при больших первых

аргументах местами были меньше значений для меньших аргументов, что говорит о зависимости алгоритма от второстепенных аргументов. Исходя из этого, для второго и третьего задания были подобраны соответствующие функции сложности. Отдельно хотелось бы выделить, что в третьем задании, исходя из измерений и опытов, один из аргументов не влияет на процесс работы программы и на результат.

1.3 Третья задача

Для решения первого задания первой задачи было решено написать программу, которая бы искала делители числа, которые образовывали частное без остатка. Приведу пример части своей программы:

A screenshot of a code editor window with a blue background. The code is written in Pascal and is enclosed in a program block labeled 's'. It uses the 'crt' unit. Variables 'n', 'i', and 'j' are declared as 'int64', while 'F' and 'G' are declared as 'text'. The program opens two files: 'C:\FPC\kek1.txt' for reading and 'C:\FPC\kek2.txt' for writing. It reads a number 'n' from the first file and then enters a loop from 'i=1' to 'i=n+1'. Inside the loop, it checks if 'n mod i = 0'. If true, it writes 'i' to the second file. After the loop, it closes both files and ends the program.

```
[■] 4
Program s;
Uses crt;
var n,i,j:int64;
    F,G:text;
begin
Clrscr;
Assign(F,'C:\FPC\kek1.txt'); ReSet(F);
Assign(g,'C:\FPC\kek2.txt'); ReWrite(G);
readln(f,n);
j:=0;
i:=1;
while i<n+1 do
begin
if n mod i = 0 then
begin
WriteLn(g,i);
end;
i:=i+1;
end;
Close(f);Close(G);
end.
```

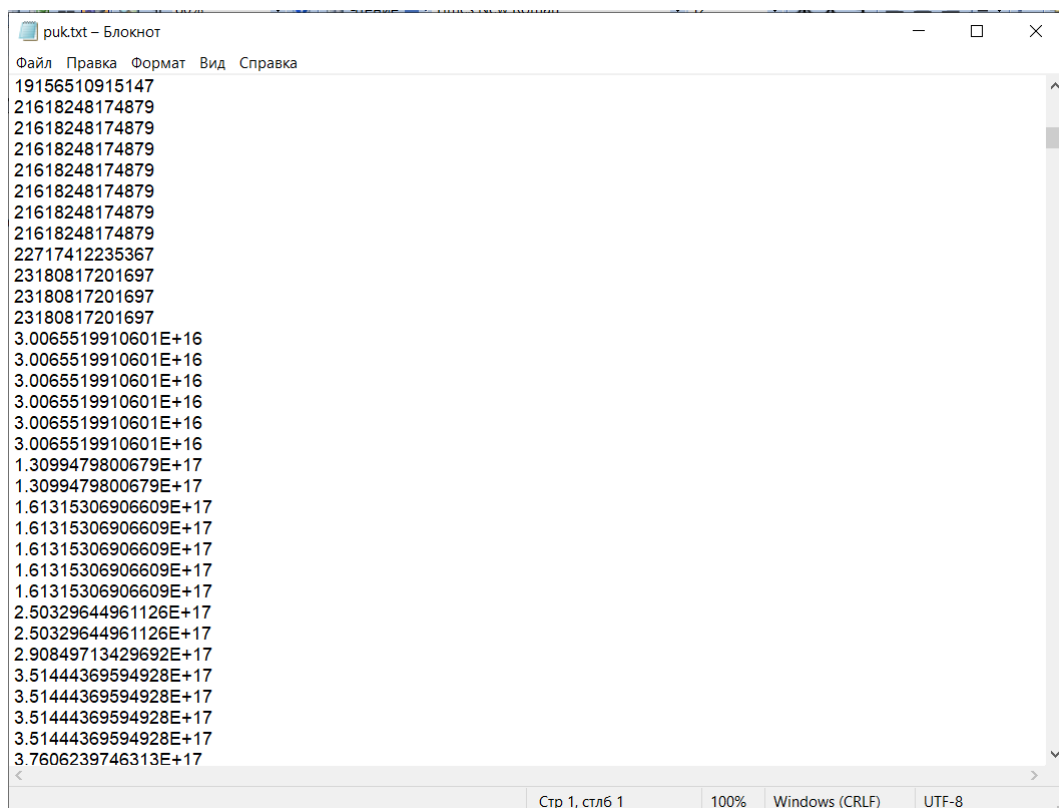
Для начала было создано два файла: один файл для чтения числа, а другой для записи в него всевозможных делителей. После проделанных действий запускался цикл, в котором отбираются делители исходного числа и записываются во второй файл. После завершения цикла файлы закрываются и, зайдя в блокнот, можно узнать количество занятых строчек и написать их в первую строчку. Общее количество делителей в результате проделанной программы получилось равным 540.

Для решения второго задания данная процедура бы не подошла, поэтому было решено для начала разбить его на пару самых небольших множителей при помощи онлайн-сервиса. После данного действия, применив правило произведения, было решено найти как можно больше делителей, составленных перемножением данных множителей. В результате была

написана ещё одна программа, которая записывала результат в отдельный файл:

```
mas[55]:= 28513;  
mas[56]:= 28513;  
assign(kek, 'C:\FPC\puk.txt');  
rewrite(kek);  
for a:=1 to 56 do  
  writeln(kek,mas[a]);  
for b:=1 to 56 do  
  writeln(kek,mas[a]*mas[b]);  
for c:=1 to 56 do  
  writeln(kek,mas[a]*mas[b]*mas[c]);  
for d:=1 to 56 do  
  writeln(kek,mas[a]*mas[b]*mas[c]*mas[d]);  
for e:=1 to 56 do  
  writeln(kek,mas[a]*mas[b]*mas[c]*mas[d]*mas[e]);  
for f:=1 to 56 do  
  writeln(kek,mas[a]*mas[b]*mas[c]*mas[d]*mas[e]*mas[f]);  
for g:=1 to 56 do  
  writeln(kek,mas[a]*mas[b]*mas[c]*mas[d]*mas[e]*mas[f]*mas[g]);  
for h:=1 to 56 do  
  writeln(kek,mas[a]*mas[b]*mas[c]*mas[d]*mas[e]*mas[f]*mas[g]*mas[h]);  
for i:=1 to 56 do  
  writeln(kek,mas[a]*mas[b]*mas[c]*mas[d]*mas[e]*mas[f]*mas[g]*mas[h]*mas[i]);  
for j:=1 to 56 do  
  writeln(kek,mas[a]*mas[b]*mas[c]*mas[d]*mas[e]*mas[f]*mas[g]*mas[h]*mas[i]*mas[j]);  
for k:=1 to 56 do  
  writeln(kek,mas[a]*mas[b]*mas[c]*mas[d]*mas[e]*mas[f]*mas[g]*mas[h]*mas[i]*mas[j]*mas[k]);  
for m:=1 to 56 do  
  writeln(kek,mas[a]*mas[b]*mas[c]*mas[d]*mas[e]*mas[f]*mas[g]*mas[h]*mas[i]*mas[j]*mas[k]*mas[m]);  
for n:=1 to 56 do  
  writeln(kek,mas[a]*mas[b]*mas[c]*mas[d]*mas[e]*mas[f]*mas[g]*mas[h]*mas[i]*mas[j]*mas[k]*mas[m]*mas[n]);  
for l:=1 to 56 do  
  writeln(kek,mas[a]*mas[b]*mas[c]*mas[d]*mas[e]*mas[f]*mas[g]*mas[h]*mas[i]*mas[j]*mas[k]*mas[m]*mas[n]*mas[l]);  
for o:=1 to 56 do  
  writeln(kek,mas[a]*mas[b]*mas[c]*mas[d]*mas[e]*mas[f]*mas[g]*mas[h]*mas[i]*mas[j]*mas[k]*mas[m]*mas[n]*mas[l]*mas[o]);  
for p:=1 to 56 do  
  writeln(kek,mas[a]*mas[b]*mas[c]*mas[d]*mas[e]*mas[f]*mas[g]*mas[h]*mas[i]*mas[j]*mas[k]*mas[m]*mas[n]*mas[l]*mas[o]*mas[p]);  
for q:=1 to 56 do  
  writeln(kek,mas[a]*mas[b]*mas[c]*mas[d]*mas[e]*mas[f]*mas[g]*mas[h]*mas[i]*mas[j]*mas[k]*mas[m]*mas[n]*mas[l]*mas[o]*mas[p]*mas[q]);  
for r:=1 to 56 do  
  writeln(kek,mas[a]*mas[b]*mas[c]*mas[d]*mas[e]*mas[f]*mas[g]*mas[h]*mas[i]*mas[j]*mas[k]*mas[m]*mas[n]*mas[l]*mas[o]*mas[p]*mas[q]*mas[r]);  
for s:=1 to 56 do  
  writeln(kek,mas[a]*mas[b]*mas[c]*mas[d]*mas[e]*mas[f]*mas[g]*mas[h]*mas[i]*mas[j]*mas[k]*mas[m]*mas[n]*mas[l]*mas[o]*mas[p]*mas[q]*mas[r]*mas[s]);
```

Главным минусом данного способа являлось то, что Free Pascal не отображает слишком большие числа, из-за чего пришлось записать лишь ограниченное количество точных делителей, хотя на самом деле их получилось около 10000.



```
puk.txt - Блокнот  
Файл Правка Формат Вид Справка  
19156510915147  
21618248174879  
21618248174879  
21618248174879  
21618248174879  
21618248174879  
21618248174879  
22717412235367  
23180817201697  
23180817201697  
23180817201697  
3.0065519910601E+16  
3.0065519910601E+16  
3.0065519910601E+16  
3.0065519910601E+16  
3.0065519910601E+16  
3.0065519910601E+16  
1.3099479800679E+17  
1.3099479800679E+17  
1.61315306906609E+17  
1.61315306906609E+17  
1.61315306906609E+17  
1.61315306906609E+17  
1.61315306906609E+17  
2.50329644961126E+17  
2.50329644961126E+17  
2.90849713429692E+17  
3.51444369594928E+17  
3.51444369594928E+17  
3.51444369594928E+17  
3.51444369594928E+17  
3.51444369594928E+17  
3.7606239746313E+17  
Стр 1, столб 1 100% Windows (CRLF) UTF-8
```

Из-за данной ограниченности удалось записать лишь около 60 делителей, что немного. При решении третьего задания не удалось придумать ничего лучше, как искать его отдельные делители и проделывать аналогичные действия, однако данный процесс занял бы много времени. Поэтому в качестве его делителей были найдены лишь число 1 и само число.

1.4 Четвертая задача

Первое задание четвёртой задачи было решено сделать при помощи дерева перебора и отсеивания. Для начала, каждое значение вершины было отсортировано. Ключом сортировки являлась вторая вершина, которая фигурировала в вершинах рёбер. Берётся одна из вершин ребра, затем идёт проверка и поиск вершин среди всех рёбер, которые совпадают с выбранной вершиной. После того, как такие рёбра были найдены, идёт запись всех значений вторых вершин рёбер, которые есть на ребре, помимо первой. После проделанных действий определяются те вершины, которые не были записаны в качестве вторых вершин рёбер. Данные действия проделываются для всех следующих вершин. В конце получается набор вершин, которые не фигурируют вместе с другими, определёнными, вершинами, в качестве вторых. Среди них остаётся выбрать те, которые дают наибольшую ценность и записать именно эти числа. Из-за разногласий во вторых вершинах может образоваться несколько групп вершин. После создания определённых групп была написана программа, позволяющая подсчитать ценность каждой группы:

```
program summ;
uses crt;
var a,g,f:text;
n,i:integer;
mas:array [1..20] of integer;
mass:array [1..20] of integer;
begin
  clrscr;
  assign(a,'C:\kek.txt');
  reset(a);
  assign(f,'C:\kekkek.txt');
  reset(f);
  assign(g,'C:\kek.w.txt');
  rewrite(g);
  n:=0;
  while not eof(a) do
  begin
    n:=n+1;
    read(a,mass[n]);
  end;
  i:=0;
  while not eof(f) do
  begin
```

После подсчёта «ценности» вершин каждой группы уже вручную выбиралась та, которая становилась главной, т.е. её «ценность» была больше «ценности» остальных.

Для подсчёта ценности вершин для второго и третьего задания были проделаны аналогичные действия. Возможно, были не учтены некоторые случаи, и задание было воспринято совсем неправильно.

1.5 Выводы по общей части

После прошедшей практики можно смело сказать, что были получены навыки, которые обязательно помогут нам в будущем при работе по специальности. Данные задачи, которые были решены в течение практики, являются задачами олимпиадного уровня, поэтому решить их было довольно сложно. Кратко хочется сказать о сложности, возникшей при решении всех четырёх задач практики. Первая задача была одной из самых лёгких, и практически никаких сложностей при решении не возникло, поэтому результат был получен аналитическим путём. Во время решения второй задачи были трудности в понимании сути поставленной цели, поэтому результаты получились слишком большие. После построения графиков, тоже были определённые проблемы с определением формулы этого графика, т.к. графики подходили сразу под несколько формул из-за своей приближенности в реальному результату. Третья задача оказалась тоже не очень сложной. При помощи простого алгоритма и программы первое задание третьей задачи не вызвала особых трудностей, а другие уже пришлось решать обходным путём. Последняя, четвертая задача, оказалось одной из самых сложных для понимания. Не особо было ясно, что понималось под рёбрами графиков. Из-за данного непонимания и возникли некоторые проблемы с решением поставленной задачи.

2 Индивидуальная часть

В данном разделе рассматриваются вопросы, связанные с выполнением индивидуального задания, выданного руководителем в рамках практики.

2.1 Формулировка решаемой задачи

Разработка графического приложения "Пятнашки"

2.2 Подходы к решению и результаты

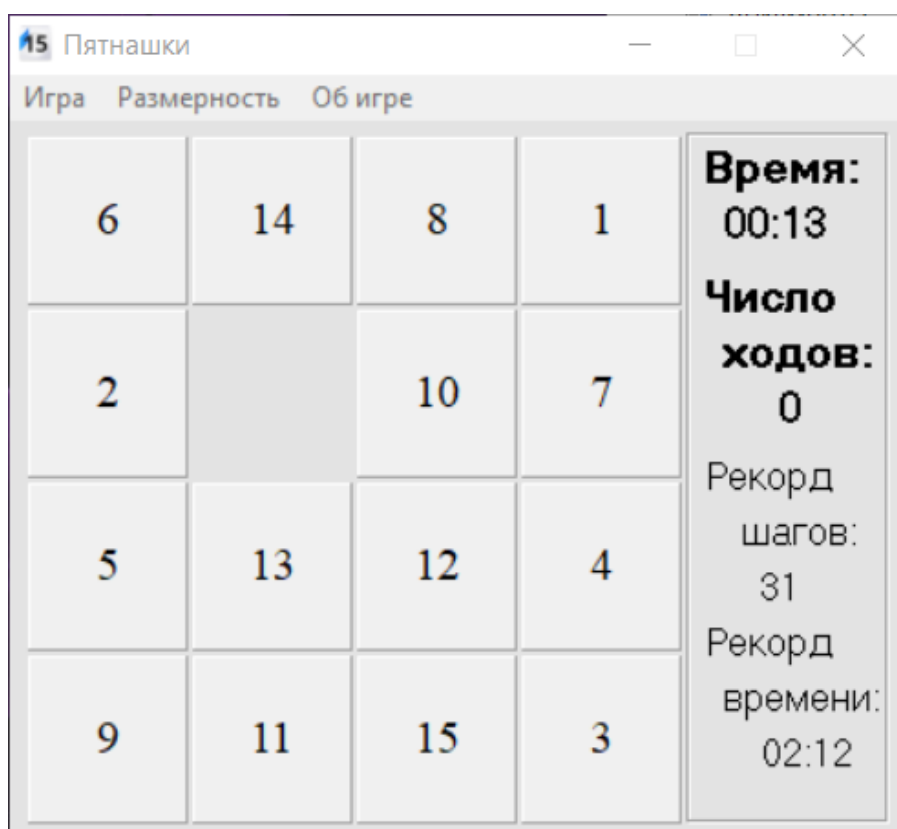
Игра «Пятнашки» - популярная головоломка, которая представлена в виде набора одинаковых квадратных костяшек с нанесёнными числами, заключёнными в квадратную коробку. Для набора из 15 элементов в коробке остаётся незаполненным одно квадратное место. Цель игры – перемещая костяшки по коробке, добиться упорядочивания их по номерам, желательнее сделав как можно меньше перемещений. Помимо обычного режима игры с полем 4x4 были реализованы поля 3x3, 6x6 и 9x9. При каждом начале новой игры или выборе новой размерности поля счётчик ходов и таймер будут обнуляться, тем самым записывая количество ходов и время, за которое пользователь прошёл данную головоломку. Также в игре предусмотрены рекорды по времени и количеству шагов, которые сделал пользователь, чтобы пройти игру.

Алгоритм действий заключается в том, чтобы менять местами пустую ячейку с той, на которую щёлкнет курсором мыши пользователь. Пустая ячейка всегда одна, соответственно, переместиться заполненная ячейка тоже может всегда лишь в одно место: вправо, влево, вниз и вверх. В моей программе данное действие было реализовано путём изменения компонента Visible на False и замене пустой ячейки на ту, у которой было изменено значение Visible. Для каждого направления своё условие. Размер самой формы, т.е. изначального квадрата, имеет константное значение – 360x400. А вот масштаб квадратиков с цифрами меняется в зависимости от выбранной размерности поля при помощи функции SetBounds, в качестве параметров которой выступают значения, специально подстраивающиеся под размерность поля. Каждая ячейка имеет свой условный номер соответствующий числу, которое на нём написано. При генерации поля данные значения перемешиваются 160 раз для создания ситуации, когда есть такие ячейки, которые не соответствуют своему условному номеру. Пользователю необходимо нажимать на ячейки до тех пор, пока они не выстроятся в верную последовательность и каждая ячейка не займёт свой

верный условный номер. После того, как пользователь завершит головоломку, выведется одно из четырёх сообщений, каждое из которых говорит об определённом завершении игры и о побитых рекордах.

После каждого перемещения ячейки пользователем счётчик ходов увеличивается на 1. Кроме подсчёта шагов также идёт таймер, который останавливается лишь тогда, когда пользователь соберёт головоломку полностью. Рекорды по времени и шагам отображаются в правой части программы и могут измениться лишь при условии, что пользователь улучшил один из них. Для этого в папке с основной программой присутствуют 2 файла, хранящих информацию о рекордах пользователя. Перед каждым запуском программы они заносятся в поля «Рекорд времени» и «Рекорд шагов», а после закрытия в эти файлы записываются новые значения, если пользователь улучшил что-то из них.

Кроме игровых возможностей также было сделано несколько косметических изменений: добавлена вкладка «Об игре» и изменён значок самой программы. Вкладка «Об игре» содержит краткое описание сути самой игры и некоторые основополагающие правила. Значок приложения был изменён на значок с символическим номером 15, который связан с названием самой игры. В качестве примера приложен снимок экрана с программой, который наглядно показывает визуальную стилистику игры:



2.3 Выводы по индивидуальной части

В ходе решения индивидуального задания были получены навыки в работе со средой Lazarus/Delphi. Для реализации некоторых возможностей программы потребовалось изучить работу с графикой в данной среде программирования. В целом, можно сказать, что благодаря индивидуальному заданию был получен необходимый опыт для работы с графикой и сделана первая игра.

Заключение

Данная практика была первой практикой по дисциплине «Программирование». Её особенностью является то, что данная практика проходила в дистанционном формате, и местом прохождения являлся «Вятский государственный университет». Индивидуальное задание и 4 практических задачи – суть всей практики.

В ходе выполнения индивидуального задания и практических заданий были получены навыки работы с графикой, сделано первое полноценное графическое приложение и составлены алгоритмы для решения практических задач.

Можно с уверенностью сказать, что первая практика окажет большое влияние в дальнейшем: при поиске работы, написании различных программ и т.д. При решении практических задач были придуманы алгоритмы, которые пригодятся не только в решении данных задач, но также помогут при составлении решения к другим классам задач. Индивидуальная часть практики помогла разобраться в работе с графикой и теперь можно пытаться писать более сложные графические приложения, придуманные лично или взятые из каких-либо источников.

В заключение хотелось бы выразить собственное мнение насчёт практики. Понравилось составлять алгоритмы и искать решения для ряда определённых задач, изучать работу некоторых приложений, но, в то же время, от практики ожидаешь чего-то большего. Хотелось бы в дальнейшем, по мере взросления и увеличению курса, проходить практику на различных предприятиях и получать опыт уже непосредственно на настоящей практике.