

УТВЕРЖДАЮ

Руководитель практики от Университета

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(должность, ФИО)*

МП \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(Подпись)*

Дата « \_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 202\_ год

**ОТЧЕТ**

**ПО УЧЕБНОЙ ПРАКТИКЕ**

ПМ.02 Осуществление интеграции программных модулей

|  |  |
| --- | --- |
| Червякова Наталья Александровна | |
| *(Ф.И.О. обучающегося)* | |
| 09.02.07 Информационные системы и программирование | |
| *(специальность)* | |
|  | |
| Учебная группа | ИСПк-201-51-00 |
|  |  |
| Место прохождения практики | ФГБОУ ВО «Вятский государственный университет», |
| Колледж ВятГУ | *(наименование организации, структурного подразделения организации)* |
|  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Итоговая оценка: |  | | | | |
| Руководитель  практики от университета |  |  |  |  |  |
|  | *(дата)* |  | *(подпись)* |  | *(Ф.И.О.)* |

Киров, 2022 г.



ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА ПРАКТИКУ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ф.И.О. обучающегося | | | Червякова Наталья Александровна | | | | | | |
| Специальность | | | 09.02.07 Информационные системы и программирование | | | | | | |
| Учебная группа | | | ИСПк-201-51-00 | | | | | | |
| Вид практики | | | учебная практика | | | | | | |
| Сроки прохождения практики с | | | | 01.09.2022 | по | 23.12.2022 | | | |
| Место прохождения практики | | | | ФГБОУ ВО «Вятский государственный университет»,  Колледж ВятГУ | | | | | |
|  | | (наименование организации, структурного подразделения организации) | | | | | | |
| № | Виды работ, выполняемых обучающимися во время практики | | | | | | Объем работ (час) | Формируемые компетенции | | |
| 1 | Пройти инструктаж по ознакомлению с правилами внутреннего трудового распорядка, охраны труда, техники безопасности, противопожарной безопасности, санитарно-эпидемиологическими правилами и гигиеническими нормативами, а также вводный инструктаж и инструктаж на рабочем месте | | | | | | 2 | - | | |
| 2 | Настройка рабочего окружения | | | | | | 6 | ОК-3, ОК-7 | | |
| 3 | Анализ заданий на разработку | | | | | | 12 | ОК-2, ПК-2.1 | | |
| 4 | Проведение исследований в заданной предметной области, поиск и анализ готовых технических решений | | | | | | 12 | ОК-1, ОК-9, ОК-11 | | |
| 5 | Реализация программного кода, встраивание кода в готовые архитектурно-структурные решения | | | | | | 36 | ОК-8, ОК-9, ОК-10, ПК-2.2, ПК-2.5, ПК-2.6 | | |
| 6 | Тестирование и отладка полученного кода | | | | | | 12 | ОК-9, ОК-10, ПК-2.3, ПК-2.4, ПК-2.7 | | |
| 7 | Подготовка отчетной документации, включая детальное описание разработанных и использованных алгоритмов | | | | | | 8 | ОК-5 | | |
| 8 | Подготовка к сдаче и сдача промежуточной аттестации | | | | | | 2 | ОК-4, ОК-6 | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Индивидуальное задание на практику разработано в соответствии с рабочей программой практики. | | | | | |
| Руководитель  практики от университета |  |  |  |  |  |
|  | (дата) |  | (подпись) |  | (Ф.И.О.) |

|  |  |
| --- | --- |
| С индивидуальным заданием ознакомлен(а) |  |
|  | (дата, подпись обучающегося) |

**ОТЗЫВ РУКОВОДИТЕЛЯ ПРАКТИКИ**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ф.И.О. обучающегося | | Червякова Наталья Александровна | | | |
| Специальность | | 09.02.07 Информационные системы и программирование | | | |
| Учебная группа | | ИСПк-201-51-00 | | | |
| Вид практики | | учебная практика | | | |
| Сроки прохождения практики с | | | 01.09.2022 | по | 23.12.2022 |
| Место прохождения практики | | | ФГБОУ ВО «Вятский государственный университет», Колледж ВятГУ | | |
|  | (наименование организации, структурного подразделения организации) | | | | |

ВИДЫ И КАЧЕСТВО ВЫПОЛНЕНИЯ РАБОТ

|  |  |  |  |
| --- | --- | --- | --- |
| Вид работ | Критерий выполнения работ | | |
| Выполнены полностью самостоятельно | Выполнены с незначительной помощью наставника | Выполнены с помощью наставника |
| Настройка рабочего окружения | V |  |  |
| Анализ заданий на разработку | V |  |  |
| Проведение исследований в заданной предметной области, поиск и анализ готовых технических решений | V |  |  |
| Реализация программного кода, встраивание кода в готовые архитектурно-структурные решения | V |  |  |
| Тестирование и отладка полученного кода | V |  |  |
| Подготовка отчетной документации, включая детальное описание разработанных и использованных алгоритмов | V |  |  |
| Подготовка к сдаче и сдача промежуточной аттестации | V |  |  |

Обучающийся ознакомлен с правилами внутреннего трудового распорядка, охраны труда, техники безопасности, противопожарной безопасности, санитарно-эпидемиологическими правилами и гигиеническими нормативами, а также прошел вводный инструктаж и инструктаж на рабочем месте.

Во время прохождения учебной практики обучающимся освоены следующие профессиональные и общие компетенции:

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование компетенции | Показатели оценки | Оценка | |
| Освоена | Не освоена |
| ПК 2.1. Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент. | Способен выполнять анализ предметной области, выявляя существенные элементы, оказывающие влияние на проектируемые решения | V |  |
| ПК 2.2. Выполнять интеграцию модулей в программное обеспечение. | Способен выполнять интеграцию собственных решений в имеющиеся программные модули | V |  |
| ПК 2.3. Выполнять отладку программного модуля с использованием специализированных программных средств. | Способен контролировать процесс выполнения созданного программного кода с использованием специализированных средств отладки | V |  |
| ПК 2.4. Осуществлять разработку тестовых наборов и тестовых сценариев программного обеспечения. | Способен выполнять оценку корректности функционирования системы | V |  |
| ПК 2.5. Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования. | Способен создавать программный код, отвечающий предъявляемым требованиям | V |  |
| ПК 2.6. Разрабатывать элементы программного модуля в соответствии с требованиями, определенными техническим заданием. | Способен создавать программный код, отвечающий предъявляемым требованиям | V |  |
| ПК 2.7. Выполнять работы по модификации программного модуля с использованием специализированных программных средств. | Способен создавать программный код, отвечающий предъявляемым требованиям | V |  |
| ОК 01. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам. | Способен анализировать специализированную информацию и находить оптимальные пути решения задач профессиональной деятельности | V |  |
| ОК 02. Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности. | Способен анализировать специализированную информацию и находить оптимальные пути решения задач профессиональной деятельности | V |  |
| ОК 03. Планировать и реализовывать собственное профессиональное и личностное развитие. | Способен определять спектр программных инструментов, позволяющих повысить личную и профессиональную результативность | V |  |
| ОК 04. Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами. | Способен конструктивно обмениваться информацией с коллегами, грамотно формулировать запросы в целях получения разъясняющей информации | V |  |
| ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста. | Способен разрабатывать проектную, техническую и пользовательскую документации | V |  |
| ОК 06. Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей. | Способен конструктивно обмениваться информацией с коллегами, грамотно формулировать запросы в целях получения разъясняющей информации | V |  |
| ОК 07. Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях. | Способен прогнозировать эффективность и ресурсозатратность используемых средств | V |  |
| ОК 08. Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности. | Способен соблюдать требования внутреннего трудового распорядка организации, охраны труда и техники безопасности в целях сохранения собственного здоровья | V |  |
| ОК 09. Использовать информационные технологии в профессиональной деятельности. | Способен применять стек современных средств разработки ПО для решения задач профессиональной деятельности в заданном контексте | V |  |
| ОК 10. Пользоваться профессиональной документацией на государственном и иностранном языке. | Способен использовать в своей работе специализированную документацию | V |  |
| ОК 11. Планировать предпринимательскую деятельность в профессиональной сфере. | Способен проектировать алгоритмические решения, принимая во внимание имеющиеся ресурсные ограничения | V |  |

**Краткая характеристика работы обучающегося**

|  |
| --- |
| Программа практики выполнена в полном объеме. Работы выполнялись с нарушением |
| сроков, с замечаниями. Была проявлена минимальная самостоятельность и умение |
| грамотно пользоваться источниками информации, находящимися в свободном доступе. |

|  |  |  |
| --- | --- | --- |
|  |  | Руководитель практики  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Подпись ФИО  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (должность)  Дата «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_\_ год |

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc89269105)

[1. СВЕДЕНИЯ О РАБОТЕ, ВЫПОЛНЕННОЙ В ПЕРИОД ПРОХОЖДЕНИЯ УЧЕБНОЙ ПРАКТИКИ 4](#_Toc89269106)

[2. НАСТРОЙКА РАБОЧЕГО ОКРУЖЕНИЯ 5](#_Toc89269107)

[3. ФОРМУЛИРОВКА ЗАДАНИЯ 6](#_Toc89269108)

[4. ОПИСАНИЕ ВЫПОЛНЕНИЯ ЗАДАНИЯ 12](#_Toc89269109)

[5. ДОПОЛНИТЕЛЬНОЕ ИНДИВИДУАЛЬНОЕ ЗАДАНИE 32](#_Toc89269110)

[ЗАКЛЮЧЕНИЕ 41](#_Toc89269111)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 42](#_Toc89269112)

# **ВВЕДЕНИЕ**

Учебная практика ПМ.02 проходила на базе Колледжа ФГБОУ ВО «Вятский государственный университет» в период с 01.09.2022 г. по 21.11.2022 г. по понедельникам.

Цель практики: сформировать у обучающихся навыки разработки алгоритмических программных модулей, которые в дальнейшем могут быть использованы как самостоятельные решение или как часть полноценного продуктового решения с применением механизмов интеграции.

Задачи практики:

– сформировать навыки анализа конкретных ограниченных задач и нахождения наиболее оптимальных решений к ним;

– закрепить полученные в ходе освоения предшествующих дисциплин умения в области разработки и отладки программного кода;

– дать представление о создании программных модулей как узкоспециализированных решениях, которые могут интегрироваться в сложные системы.

# **1. СВЕДЕНИЯ О РАБОТЕ, ВЫПОЛНЕННОЙ В ПЕРИОД ПРОХОЖДЕНИЯ УЧЕБНОЙ ПРАКТИКИ**

В период с 11.01.2022 по 16.06.2022 при прохождении учебной практики ПМ.02 на базе Колледжа ФГОБУ ВО «Вятский государственный университет» был выполнен следующий перечень работ, представленный в таблице 1.

Таблица 1 – Сведения о работе, выполненной в период практики

| Дата | Краткое содержание выполненных работ |
| --- | --- |
| 06.09.2022 | Ознакомление с правилами внутреннего трудового распорядка, охраны труда, техники безопасности, противопожарной безопасности, санитарно-эпидемиологическими правилами и гигиеническими нормативами, а также вводный инструктаж и инструктаж на рабочем месте |
| 06.09.2022-29.11.2022 | Настройка рабочего окружения |
| 06.09.2022-29.11.20212 | Анализ заданий на разработку |
| 06.09.2022-13.12.2022 | Проведение исследований в заданной предметной области, поиск и анализ готовых технических решений |
| 06.09.2022-29.11.2022 | Реализация программного кода, встраивание кода в готовые архитектурно-структурные решения |
| 06.09.2022-29.11.2022 | Тестирование и отладка полученного кода |
| 06.12.2022-20.12.2022 | Подготовка отчетной документации, включая детальное описание разработанных и использованных алгоритмов |
| 20.12.2022 | Подготовка к сдаче и сдача промежуточной аттестации |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата) (подпись)

# **2. НАСТРОЙКА РАБОЧЕГО ОКРУЖЕНИЯ**

В ходе прохождения практики был использован следующий набор программных средств.

**PascalABC.NET** – это свободно распространяемая интегрированная среда разработки (IDE) и компилятор для языка программирования Pascal. Она предоставляет удобный интерфейс для создания, отладки и выполнения программ на Pascal.

Основные особенности PascalABC.NET:

* Бесплатная и открытая для использования.
* Разработана на базе компилятора Free Pascal, что обеспечивает высокую скорость выполнения программ.
* Поддерживает популярные операционные системы, такие как Windows, Linux и macOS.
* Имеет удобный визуальный редактор форм для создания графических пользовательских интерфейсов.
* Поддерживает различные стандарты и библиотеки языка Pascal, включая Delphi-совместимые библиотеки.
* Обладает мощной системой отладки, позволяющей отслеживать ошибки в коде и устранять их.
* Поддерживает различные режимы компиляции, в том числе режимы отладки и оптимизации кода.
* Позволяет экспортировать программы в различные форматы, включая исполняемые файлы и библиотеки.
* Имеет обширную документацию и сообщество пользователей, которые готовы помочь в решении возникающих вопросов.

PascalABC.NET является одной из самых популярных сред разработки на языке Pascal и широко используется в образовательных учреждениях для обучения программированию.

Интерфейс PascalABC.NET представлен на рисунке 2.1

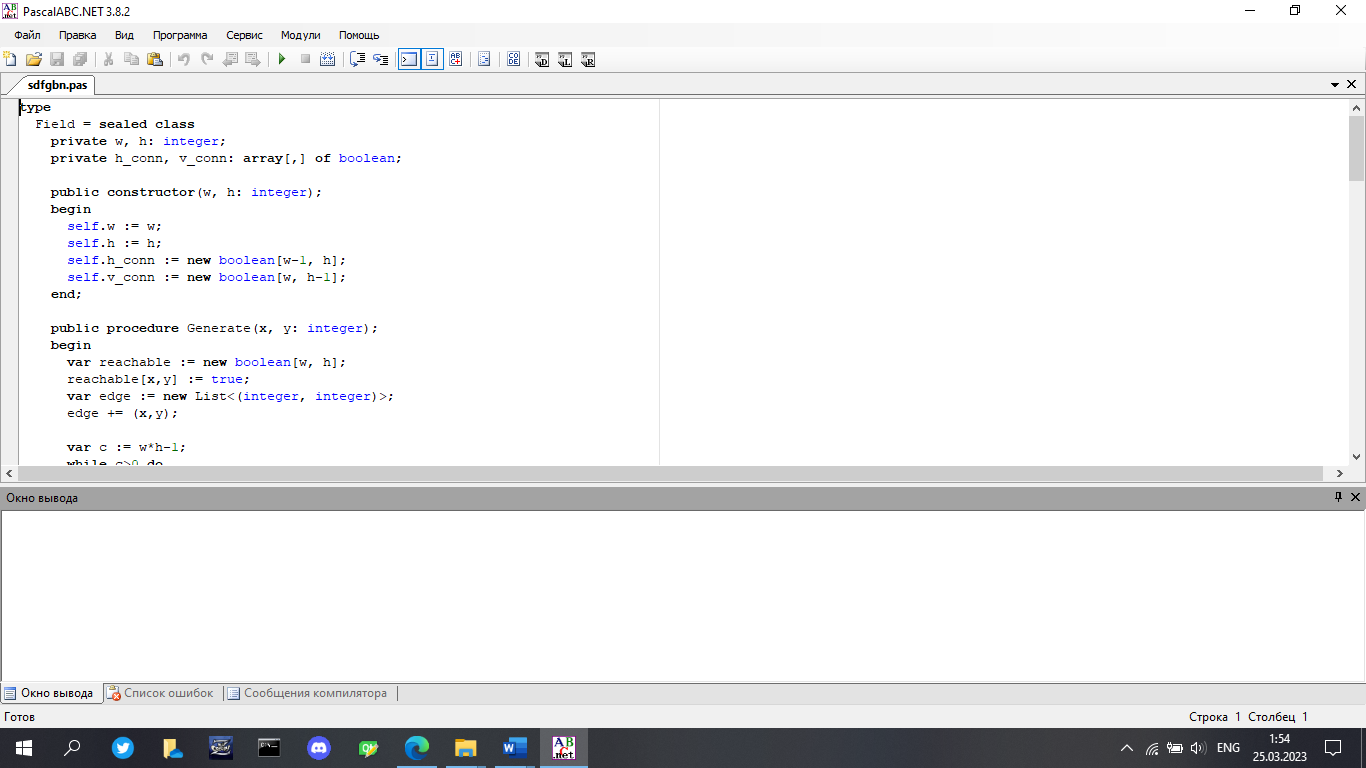


Рисунок 2.1 – PascalABC.NET

**Visual Studio Code (VS Code)** – это бесплатная кросс-платформенная интегрированная среда разработки (IDE), разработанная компанией Microsoft. Она поддерживает множество языков программирования и используется для разработки приложений как для веб-браузеров, так и для настольных операционных систем.

Ниже представлены некоторые основные характеристики Visual Studio Code:

* Расширяемость: VS Code имеет множество расширений, которые позволяют расширить его функциональность, добавить поддержку других языков программирования, инструментов, тем и многое другое.
* Отладка: Встроенная отладка позволяет выполнять отладку кода внутри VS Code, что значительно упрощает процесс нахождения и исправления ошибок в коде.
* Автодополнение: VS Code предоставляет подсказки при написании кода, что позволяет ускорить процесс разработки и уменьшить количество ошибок.
* Поддержка множества языков программирования: VS Code поддерживает множество языков программирования, включая JavaScript, Python, C#, Java, PHP, Ruby, и многие другие.

В целом, Visual Studio Code является мощной и гибкой IDE, которая может быть использована для разработки различных типов приложений и поддерживает множество языков программирования.

Интерфейс Visual Studio Code представлен на рисунке 2.2.

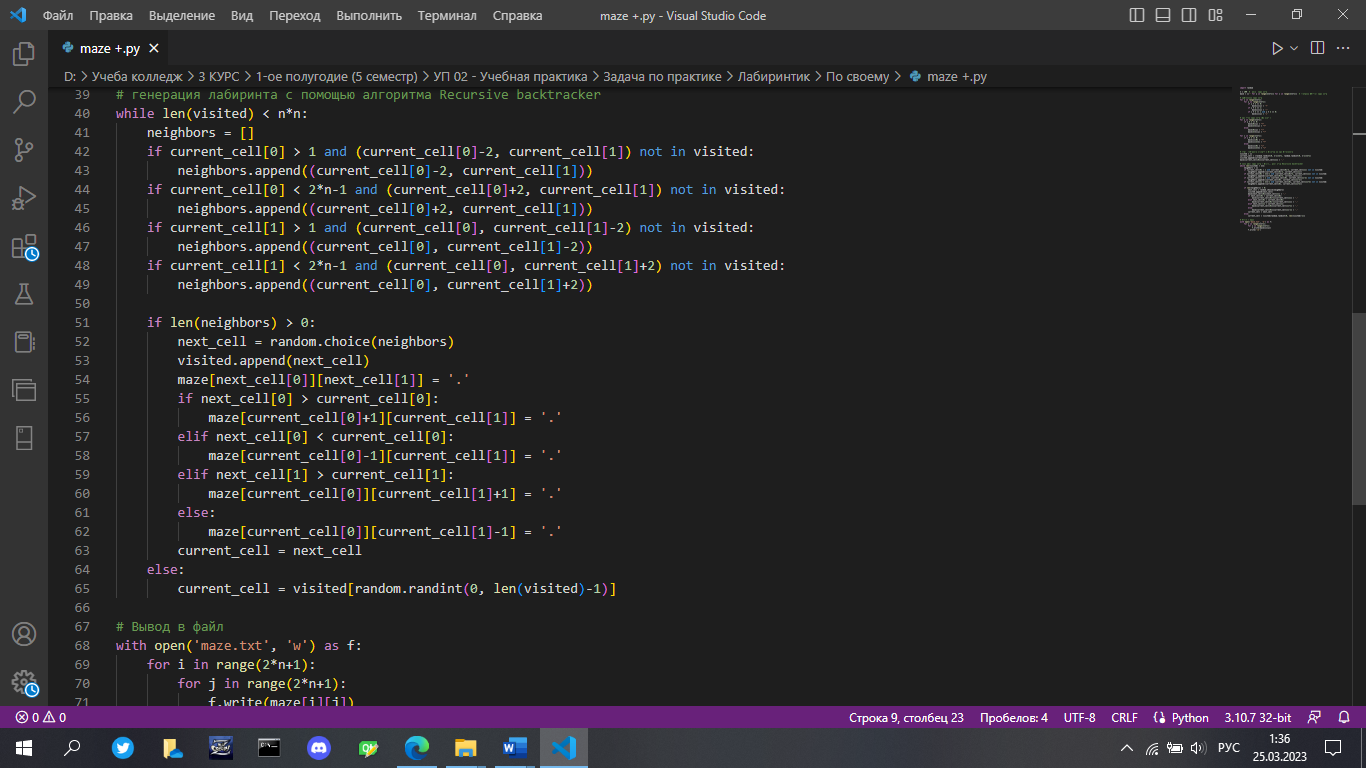


Рисунок 2.2 – IDE Visual Studio Code

**3. ФОРМУЛИРОВКА ЗАДАНИЯ**

В соответствии с заданием на учебную практику на протяжении семестра выполнялось решение следующих узконаправленных задач.

1. **Квадраты**

В указанном графическом файле находится изображение, состоящее из нескольких черных квадратов. Размер каждого квадрата равен 20 на 20 пикселей, и их расположение в изображении является псевдослучайным. Требуется провести максимально точный анализ и определить количество квадратов, находящихся на данном изображении.

1. **Несвязное множество**

Дан некоторый неориентированный вершинно-взвешенный граф, содержащий набор вершин и ребер. Цель состоит в выборе набора вершин из заданного графа, который не связан между собой, но имеет максимальную возможную суммарную весовую ценность.

1. **Делители**

Каждый файл содержит одно число с определенным набором делителей. Требуется максимизировать количество положительных делителей данного числа. Необходимо выявить и перечислить найденные делители вместе с их количеством.

1. **Задача о назначениях**

Существует некоторое количество предприятий, которые могут находиться в различных локациях, и для определения стоимости расположения этих предприятий используется матрица, в которой строки соответствуют предприятиям, а столбцы – локациям. Каждый элемент матрицы обозначает стоимость расположения конкретного предприятия в конкретной локации. Кроме того, для описания объема перевозок задана матрица, элементы которой обозначают количество грузов, и матрица стоимости перевозки, где каждый элемент обозначает стоимость перевозки груза между двумя локациями. Цель заключается в распределении производства по локациям с целью минимизации суммарной стоимости размещения всех предприятий и перевозок между ними.

1. **Раскраска графа**

Для данного неориентированного графа, представленного в виде набора вершин и ребер, каждая вершина обладает определенным "цветом", который представляется числом натурального ряда. Цель заключается в том, чтобы правильно раскрасить граф, используя как можно меньше цветов. Корректной раскраской графа является такое отображение вершин на цвета, что каждой вершине соответствует только один цвет, и соседние вершины не имеют одинаковый цвет.

1. **Пятнашки**

Задача состоит в том, чтобы решить головоломку, состоящую из n квадратных костяшек, каждая из которых имеет число, расположенное на матрице n на n. В матрице имеется одно пустое место, куда можно перемещать только костяшки, расположенные ближе всего к пустому месту. Цель состоит в том, чтобы переместить костяшки таким образом, чтобы достичь правильной расстановки чисел с наименьшим количеством ходов.

1. **Бинарная матрица**

Задача заключается в нахождении прямоугольной подматрицы наибольшей площади в заданной матрице, где все элементы состоят либо только из нулей, либо только из единиц. Для решения задачи необходимо произвести перебор всех возможных прямоугольных подматриц, сравнить их площади и выбрать максимальную. Результатом является указание координат левого верхнего и правого нижнего элементов найденной прямоугольной подматрицы.

1. **Разбиение на пары**

Дано множество натуральных чисел, которые попарно не повторяются. При выборе нескольких пар из этого множества, таких, что каждый элемент входит не более чем в одну пару, можно определить общий вес этой совокупности пар как сумму весов каждой из пар. Необходимо выполнить корректное разбиение некоторого подмножества элементов на пары таким образом, чтобы общий вес совокупности пар был максимальным.

1. **3-разбиение**

Для максимизации количества выполнимых дизъюнктов из набора трех переменных дизъюнкций необходимо определить значения всех переменных в множестве. Дизъюнктом называется набор логических переменных или их отрицаний, объединенных связкой "ИЛИ". Дизъюнкт считается выполнимым, если его истинное значение может быть получено. В качестве обозначения истинного значения переменной используется символ "1", а для ложного значения - "0". Также задано конечное множество переменных, для которых определен набор трех переменных дизъюнкций. Необходимо указать значения переменных в порядке, заданном в множестве.

1. **Неизвестный алгоритм**

Представлена определенная программа, которая выполняет алгоритм. Скорость выполнения алгоритма зависит от набора чисел, используемых в качестве входных данных. Необходимо получить максимально точное определение средней асимптотической оценки, которая определяется временем выполнения алгоритма в программе.

1. **Обобщенная игра «Instant Insanity»**

Предоставлено конечное множество красок и множество кубиков, содержащее равное количество элементов. Каждый кубик имеет несколько граней, каждая из которых окрашена в цвет из множества красок. Требуется повернуть каждый кубик таким образом, чтобы, располагая их в вертикальном столбике, каждый цвет из множества красок встречался как можно меньшее количество раз. Необходимо представить ответ в виде расположения кубиков в столбике.

1. **Прямоугольное сжатие**

Для заданной матрицы, содержащей только элементы 0 и 1, требуется разбить ее на прямоугольные подматрицы, содержащие только 0 или только 1. Цель заключается в том, чтобы минимизировать количество полученных подматриц. В ответе следует указать число полученных подматриц и перечислить каждую прямоугольную подматрицу, определив ее левый верхний элемент (его строку и столбец) и правый нижний элемент (его строку и столбец).

# **4. ОПИСАНИЕ ВЫПОЛНЕНИЯ ЗАДАНИЯ**

В данном разделе подробно рассмотрены использованные подходы к решению четырех задач практики: пятнашки, бинарная матрица, обобщенная игра Instant Insanity, прямоугольное сжатие.

**4.1 Пятнашки**

В этой задаче необходимо переместить костяшки таким образом, чтобы достичь правильной расстановки чисел с наименьшим количеством ходов.

Для решения данной задачи использовался сайт (см. Список использованных источников), позволяет пользователю установить ячейки в нужном порядке и предоставляет информацию о правилах игры в "Пятнашки". Интерфейс сайта представлен на рисунке 4.1.1

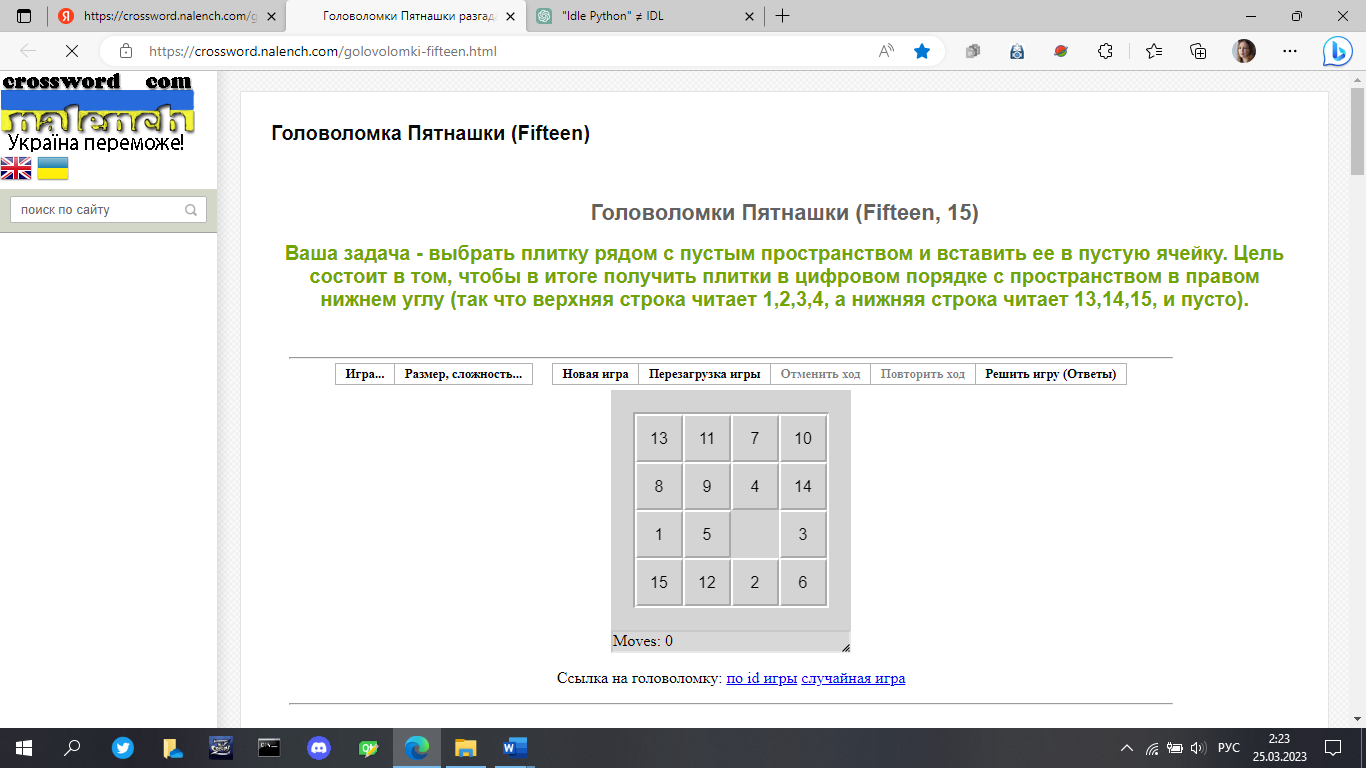


Рисунок 4.1.1 – Интерфейс сайта «Головоломки Пятнашки»

Во второй кнопке «Размер, сложность…» можно задать любую соответствующую размерность поля при этом указав соответствующую длину и ширину. Введение размерности поля для пятнашек представлен на рисунке 4.1.2.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 4.1.2 – Размерность поля

Далее, проанализировав некоторые функции данного сайта, через опцию "Собственное id игры" можно последовательно ввести все необходимые числа в строку перечисления ячеек, следуя примеру, показанному на рисунке 4.1.3.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4.1.3 – Ввод чисел в строку перечисления ячеек

Для выполнения задания было создано игровое поле на сайте, которое отображается в нужном порядке. Нулём обозначается пустая ячейка. Изображение игрового поля представлено на рисунке 4.1.4.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4.1.4 – Сгенерированное игровое поле

После того, как головоломка была завершена и числа были упорядочены по порядку от 1 до конечного числа, пустая ячейка находится в самом конце, все ходы были посчитаны. Затем, необходимо было записать количество всех ходов в первой строке и сопоставить все числа между собой с учетом этих ходов.

Завершенная головоломка представлена на рисунке 4.1.5

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4.1.5 – Пройденная игра «Пятнашки»

Для получения ответа необходимо записать последовательность координат передвигаемой ячейки. Каждая пара координат записывается на новой строке, где первое число – номер строки, а второе – номер столбца. Общее количество ходов указывается в первой строке файла.

Для формирования ответа была использована программа Word. В заданный документ были вставлены все скриншоты ходов костяшек, а затем высчитывалась последовательность координат передвигаемой ячейки по каждому скриншоту.

Подсчёт последовательности координат передвигаемой ячейки представлен на рисунке 4.1.6 и 4.1.7

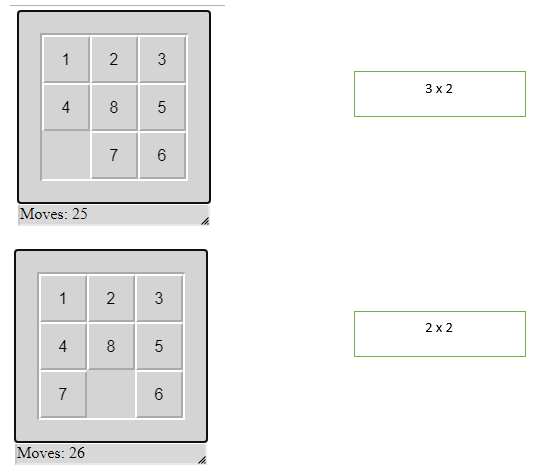
****

Рисунок 4.1.6 – Подсчёт результата task1

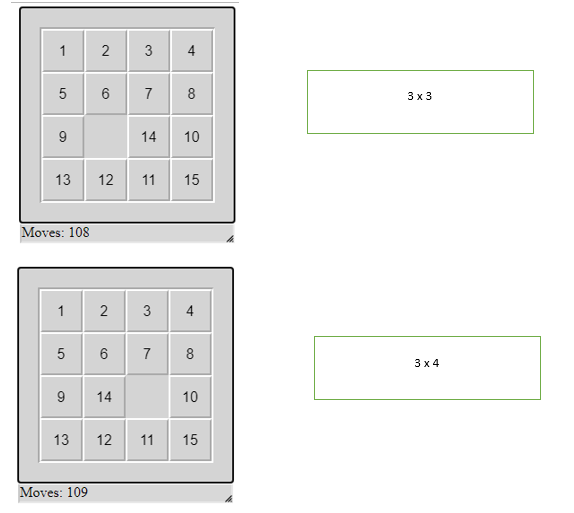
****

Рисунок 4.1.7 – Подсчёт результата task2

Task3 может быть решен аналогичным способом, однако вычисление результата может занять слишком много времени, что не позволит выполнить задание в срок. Хотя можно было бы написать программу на языке Python для решения этой задачи, но, к сожалению, для реализации такой программы не хватает необходимых знаний.

**4.2 Бинарная матрица**

В задаче необходимо найти в заданной матрице максимальную по площади прямоугольную подматрицу, состоящую только из нулей или только из единиц. Для решения задачи необходимо произвести перебор всех возможных прямоугольных подматриц, сравнить их площади и выбрать максимальную.

Для решения данной задачи можно было использовать программу Microsoft Visio представлена на рисунке 4.2.1. Однако, для решения второй и третьей задачи, такой подход не подходит, так как он не позволяет определить матрицу и вычислить результаты. В этом случае можно воспользоваться Microsoft Excel, но моя архитектура ноутбука не позволяет обработать большие объемы данных при вставке в таблицу

Максимальную подходящую матрицу можно было также определить в программе "Блокнот" с помощью визуального анализа, но при этом неудобно будет вычислять результат для каждой подматрицы из-за уменьшенного масштаба текстового документа.

Данная подматрица, которая выделена в красную рамочку изображена на Рисунке 4.2.2.

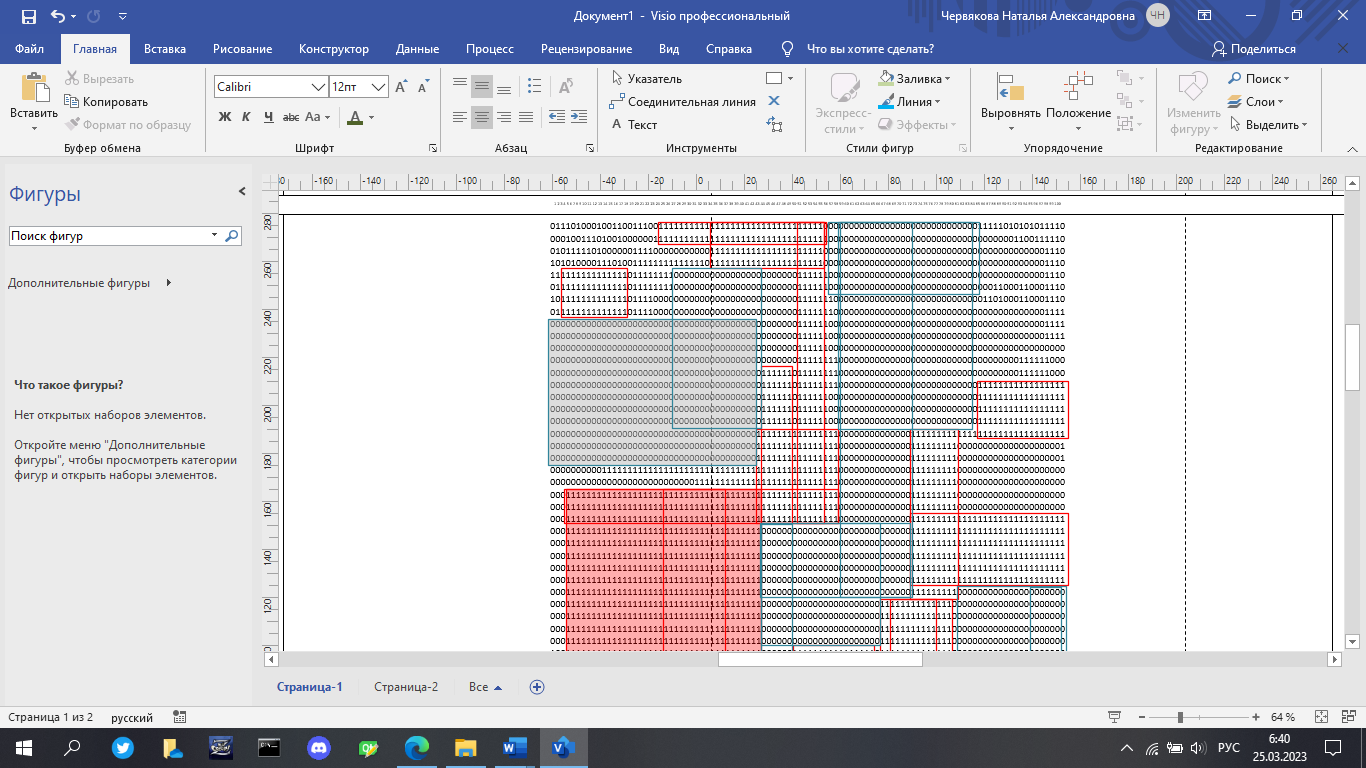


Рисунок 4.2.1 – Решение task1 через Microsoft Visio

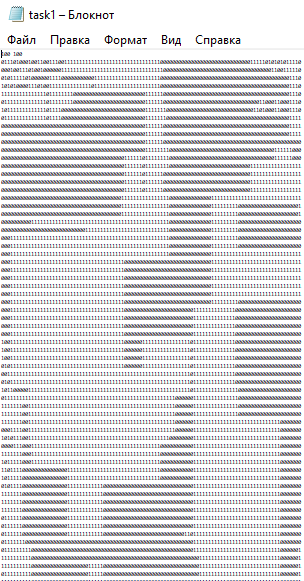
****

Рисунок 4.2.2 – Визуальный анализ нахождения подматрицы в файле task1

Полное решение task1 в Microsoft Visio представлен на рисунке 4.2.3.

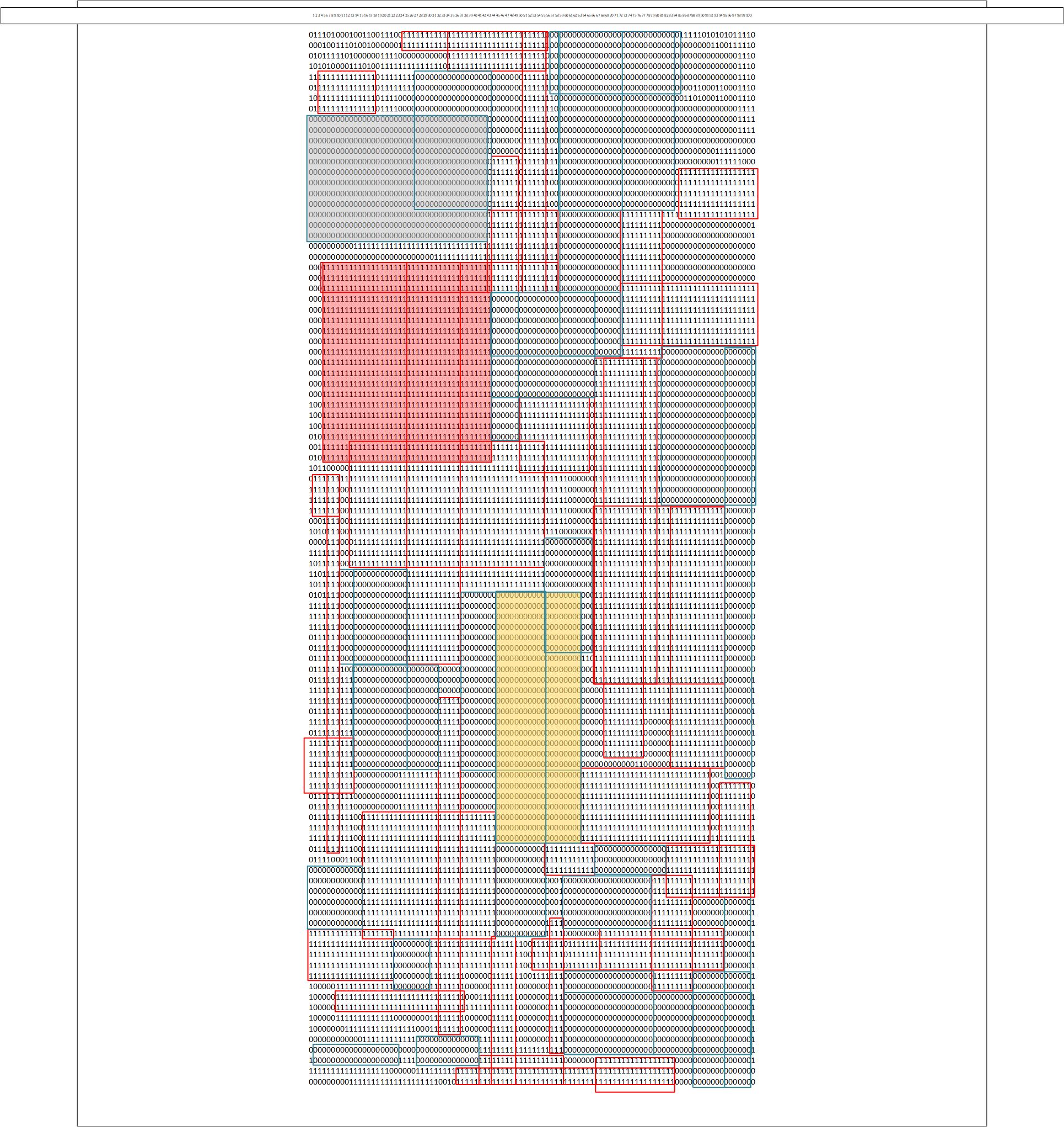


Рисунок 4.2.3 – Решение task1

Подматрицы, содержащие только элементы со значением 1, выделены красной рамкой, а подматрицы, содержащие только элементы со значением 0, выделены синей рамкой. Данные выделенные подматрицы представлены на рисунке 4.2.4 и 4.2.5.

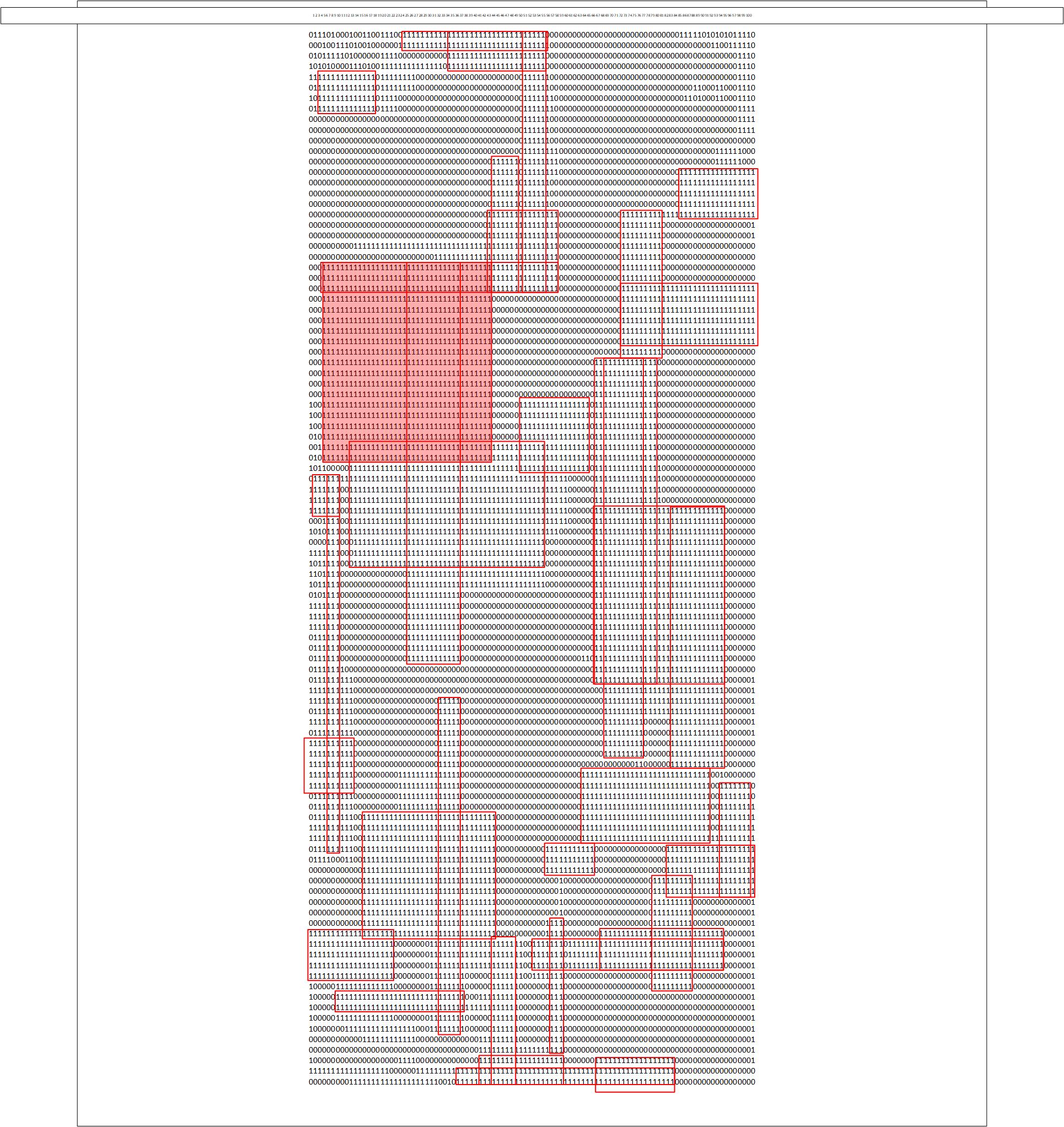


Рисунок 4.2.4 – Выделение подматриц со значением 1

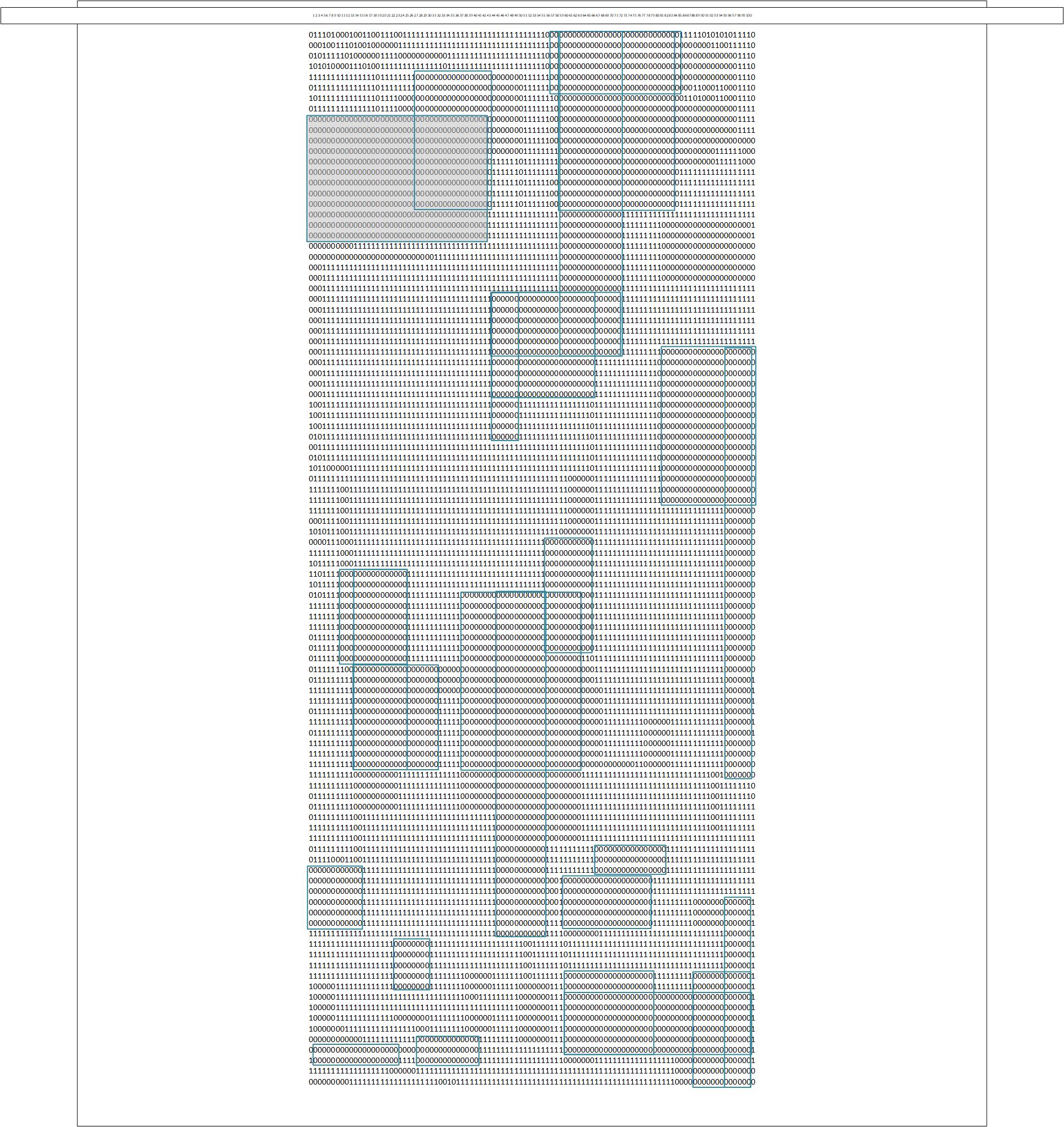


Рисунок 4.2.5. – Выделение подматриц со значением 0

После того, как на рисунке 4.2.3 были выделены наибольшие подматрицы, они были закрашены цветом (синий, красный, желтый на рисунке 4.2.6), а затем определялась самая большая из них, представлены на рисунке 4.2.7.



Рисунок 4.2.6 – Наибольшие закрашенные подматрицы

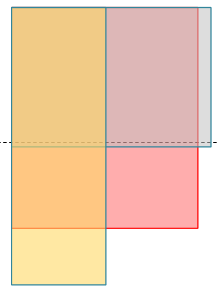


Рисунок 4.2.7 – Нахождение наибольшей подматрицы

Под красной заливкой оказалась самая большая подматрица, которая представлена на рисунке 4.2.8.

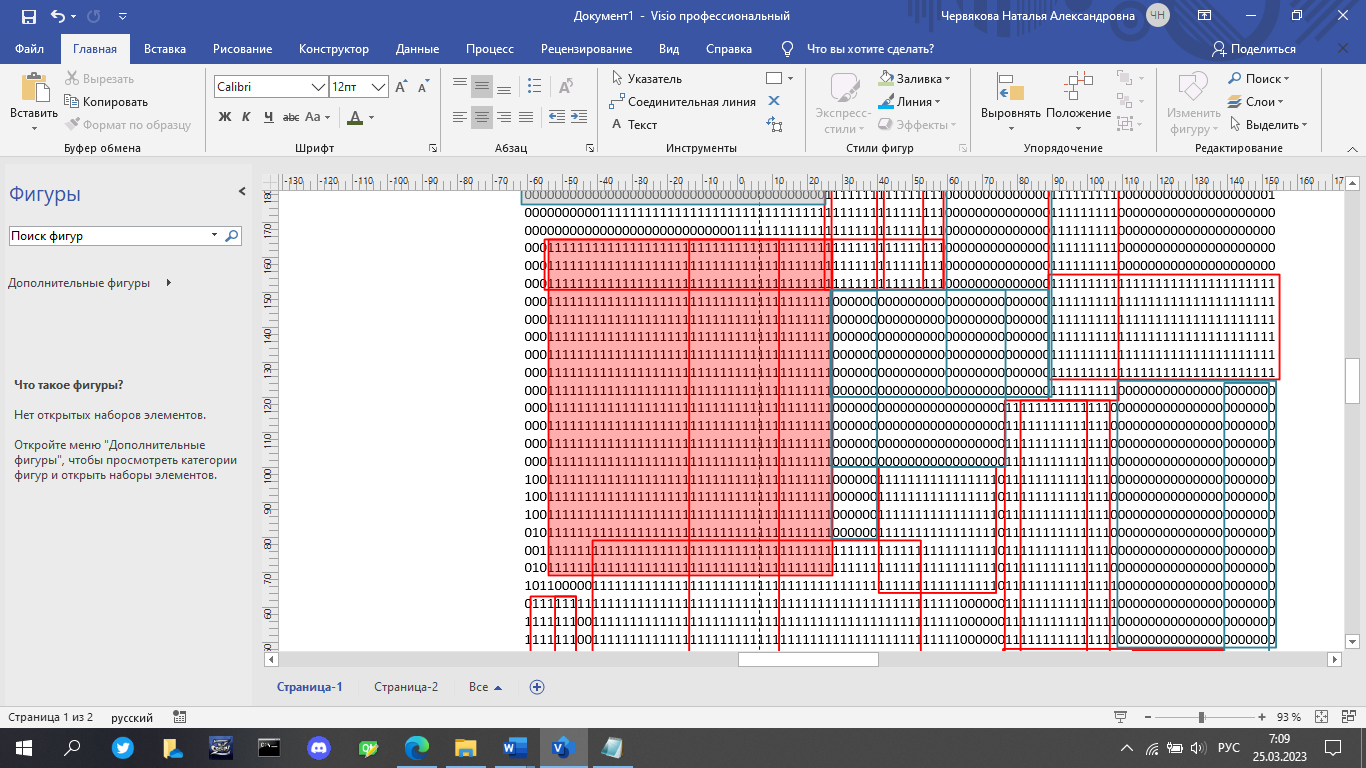


Рисунок 4.2.8 – Итоговая подматрица для ответа

После обнаружения подматрицы итоговый результат записываем в текстовый файл, где указываем номер верхней строки, номер левого столбца, номер нижней строки и номер правого столбца.

**4.3 Обобщенная игра Instant Insanity**

Предоставлено конечное множество красок и множество кубиков, содержащее равное количество элементов. Каждый кубик имеет несколько граней, каждая из которых окрашена в цвет из множества красок. Требуется повернуть каждый кубик таким образом, чтобы, располагая их в вертикальном столбике, каждый цвет из множества красок встречался как можно меньшее количество раз.

Для решения данной задачи использовалась программа Paint 3D, программа представлена на рисунке 4.3.1. Данная программа имеет трёхмерную фигуру куба, что упрощает ручное рисование, и также предоставляет возможность изменять его положение, поворачивать, рисовать на нём и изменять его цвет (Рисунок 4.3.2). Сначала было не ясно, как правильно пронумеровать грани по развертке куба, но для упрощения процесса создания был создан небольшой макет куба в реальном мире, и грани были пронумерованы цифрами. Это позволило впоследствии легче наносить цифры на трёхмерную фигуру куба в программе.

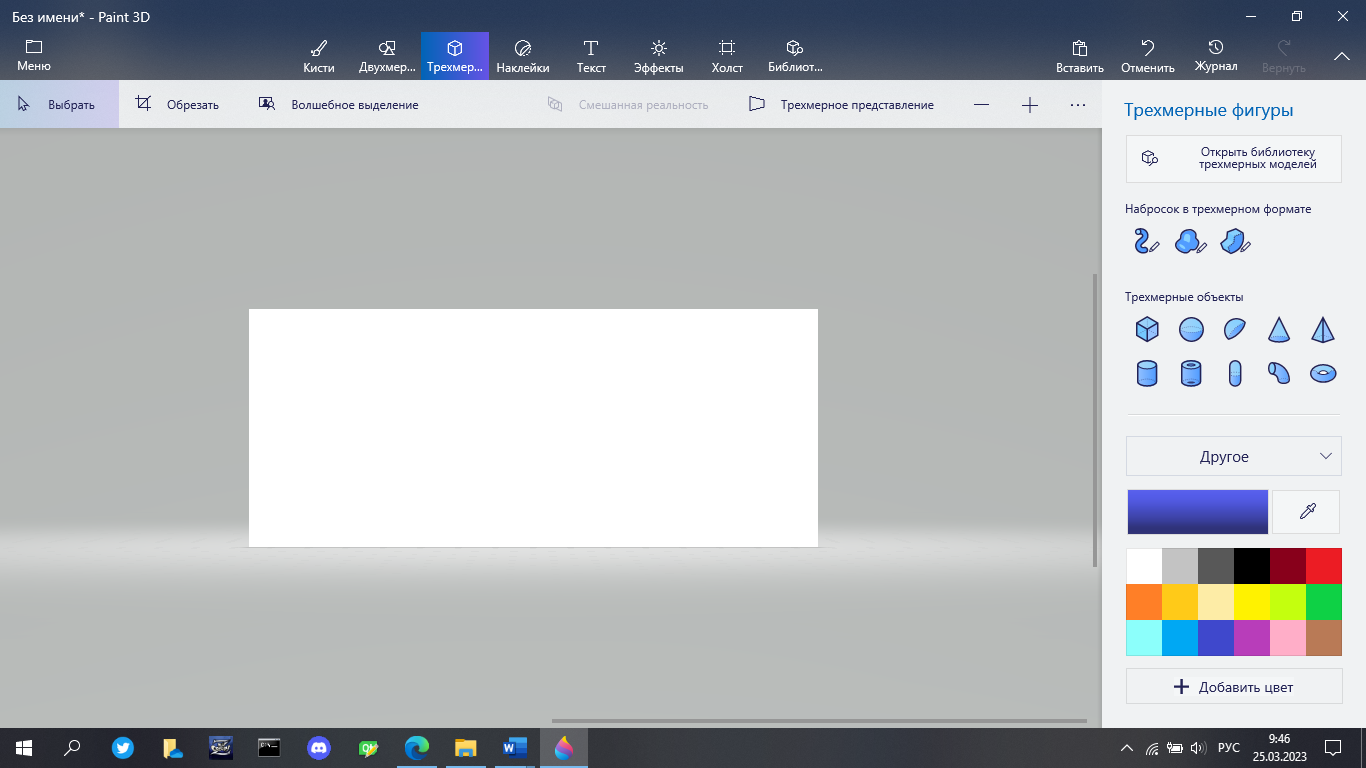


Рисунок 4.3.1 – Paint 3D

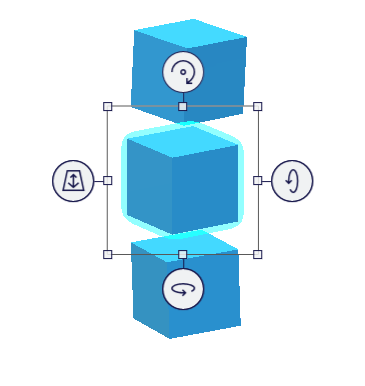


Рисунок 4.3.2 – Трехмерные фигуры в Paint 3D

Для выполнения задачи необходимо поочередно расположить каждый кубик сверху вниз на макете и повернуть его таким образом, чтобы на стороне, противоположной выбранному цвету из множества C, находился тот цвет из C, который реже всего встречается на вертикальном столбике.

Если все цвета из C встречаются на каждой стороне вертикального столбика одинаково часто, то необходимо повернуть кубик таким образом, чтобы на стороне, противоположной выбранному цвету из C, был любой цвет, кроме того, который уже присутствует на этой стороне.

После каждого поворота необходимо перепроверять расположение цветов из C на вертикальном столбике до тех пор, пока не будет достигнуто правильное расположение.

Результат решения task1 представлен на рисунке 4.3.3.

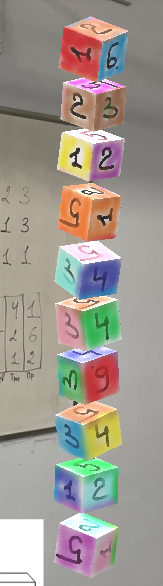


Рисунок 4.3.3 – Решение task1

Решение task2 основывается на том же методе, что и в task1. Однако ручное решение task2 и task3 через Paint 3D представляется трудоемким процессом, поскольку при вращении кубиков и изменении их цвета легко запутаться. Кроме того, из-за ограниченного времени на решение задач ручным методом не всегда удастся успеть.

**4.4 Прямоугольное сжатие**

Для заданной матрицы, содержащей только элементы 0 и 1, требуется разбить ее на прямоугольные подматрицы, содержащие только 0 или только 1.

Для решения данной задачи было использовано программное обеспечение, включающее Microsoft Visio, с помощью которого необходимо было разбить исходную матрицу на наименьшее количество подматриц, состоящих только из нулей или единиц. Решение task1 и task2 возможно при использовании этого подхода. Однако, task3 также может быть решен, но на выполнение выделено только одни сутки. Можно написать программу для разбиения матрицы, чтобы справиться с задачей вовремя, но неясно, каким образом можно реализовать данную программу.

Результат решения task1 представлен на рисунке 4.4.1.



Рисунок 4.4.1 – Решение task1

После выделения наибольших по размеру прямоугольников и квадратов, были выделены также средние с прямоугольной и квадратной рамкой.  
 Матрицы, содержащие только элементы со значением 1, выделены синим прямоугольником и квадратом, а матрицы, содержащие только элементы со значением 0, выделены красным прямоугольником и квадратом.

Выделенные матрицы представлены на рисунке 4.4.2 и на рисунке 4.4.3

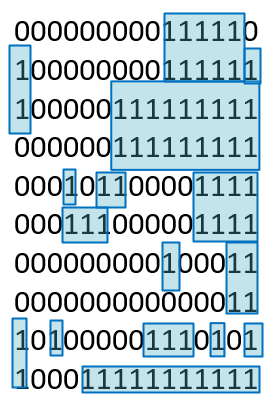


Рисунок 4.4.2 – Выделение элементов со значением 1

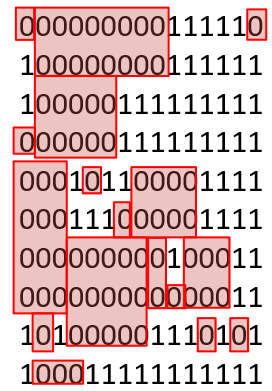


Рисунок 4.4.3 – Выделение элементов со значением 0

Для удобства, была сделана специальная рамка, которая будет показывать номер строки и номер столбца, так как нумерация столбцов и строк в ответе начиналась с 0, а не с 1. Рамка представлена на рисунке 4.4.4.



Рисунок 4.4.4 – Рамка для записывания ответа

Нули или единицы, которые остались, были разбиты на пары и выделены, а оставшиеся одиночные подматрицы были просто записаны в ответ.

При записи ответа были найдены более оптимальные подматрицы.

Результат решения task2 представлен на рисунке 4.4.5.

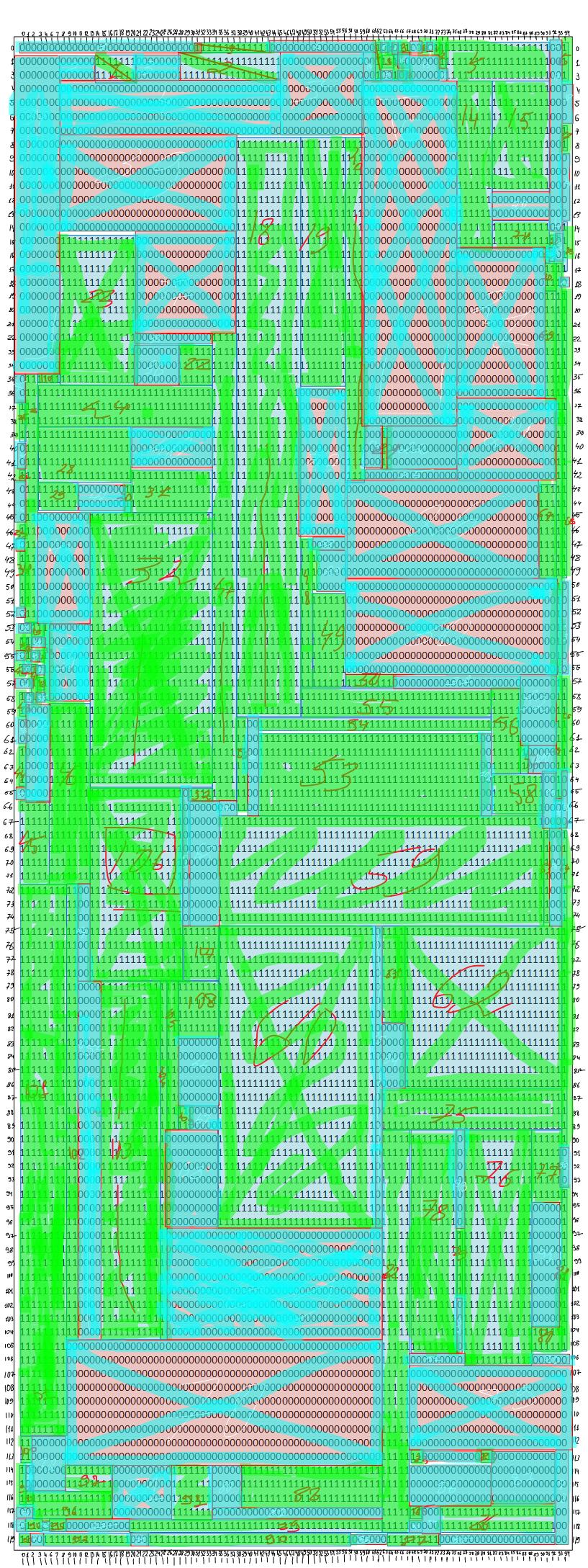


Рисунок 4.4.5 – Решение task2

Для того, чтобы решить task2, используется тот же метод решения, что и в task1.

В этом формате были закрашены большие подматрицы, затем подматрицы приближала и проверяла на отсутствие единиц или нулей. Как только был закрашен достаточно большой объем, переходила к более мелким подматрицам.

# **5. ДОПОЛНИТЕЛЬНОЕ ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ**

В связи с наличием замечаний в ходе осуществления работ, в период практики выполнялось дополнительное задание. В рамках данного задания требовалось решить задачу лабиринт.

Основная суть задачи заключается в том, что необходимо сгенерировать лабиринт, состоящий из 100 на 100 комнат, которые будут связаны между собой коридорами таким образом, чтобы существовал только один путь между любыми двумя комнатами.

Необходимо гарантировать корректную генерацию лабиринта (как показано на Рисунке 5.1), чтобы решение задачи было правильным и не было отвергнуто преподавателем (как показано на Рисунке 5.2).

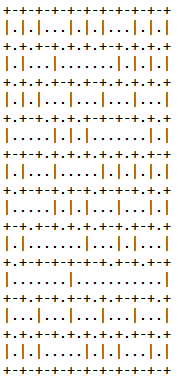


Рисунок 5.1 – Пример корректного лабиринта

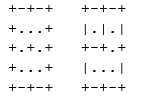


Рисунок 5.2 – Примеры некорректного лабиринта

Основные правила формирования лабиринта:

* лабиринт n×n состоит из (2n+1) строк и (2n+1) столбцов;
* пронумеруем строки от 1 до (2n+1) сверху вниз, столбцы – справа налево;
* если клетка имеет нечетный номер строки и нечетный номер столбца, ставим ’+’;
* если клетка имеет четный номер строки и четный номер столбца, ставим ’.’ – это комната;
* если клетка имеет четный номер строки и нечетный номер столбца, ставим ’.’ или ’|’ – это горизонтальных проход между комнатами (открытый и заблокированный соответственно);
* если клетка имеет нечетный номер строки и четный номер столбца, ставим ’.’ или ’-’ – это вертикальный проход между комнатами (открытый и закрытый соответственно);
* на периметре лабиринта не может быть открытых проходов.

Для написания кода задачи была рассмотрена возможность использования IDE Visual Studio Code, однако из-за постоянных зависаний и долгой откладкой кода было принято решение воспользоваться более простым вариантом – IDLE Python (Рисунок 5.3), который работает быстро и не вызывает зависаний.

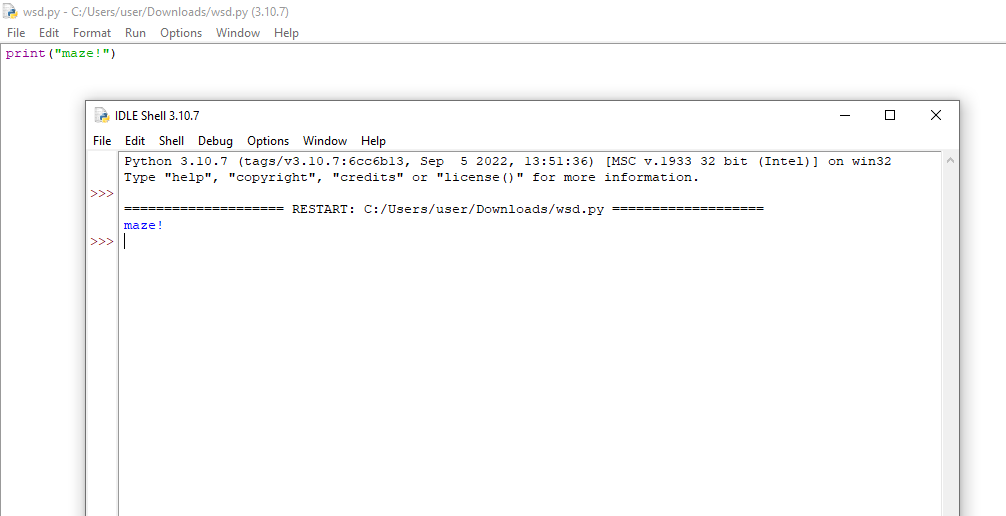


Рисунок 5.3 – IDLE Python

Для решения задачи генерации лабиринта был использован алгоритм Recursive backtracker, также известный как поиск в глубину. Этот алгоритм является более подходящим для решения данной задачи.

Немного теории о самом алгоритме:

Алгоритм Recursive backtracker – это один из алгоритмов генерации лабиринта, который использует поиск в глубину. Этот алгоритм начинает с одной ячейки и строит лабиринт постепенно, случайным образом выбирая соседнюю ячейку, которая еще не была посещена.

Алгоритм затем рекурсивно перемещается в выбранную ячейку и продолжает строить лабиринт из нее, выбирая новую не посещенную соседнюю ячейку. Если в текущей ячейке больше нет не посещенных соседей, то алгоритм возвращает управление к предыдущей ячейке в стеке рекурсии и продолжает строить лабиринт из нее.

Этот алгоритм генерирует лабиринты, которые имеют множество тупиков, что может привести к тому, что поиск решения лабиринта может быть сложным. Однако, Recursive backtracker является достаточно простым и эффективным алгоритмом для генерации лабиринтов.

**Листинг программы:**

import random

n = 100 # размер лабиринта

maze = [['' for j in range(2\*n+1)] for i in range(2\*n+1)] # создание пустого лабиринта

# заполнение лабиринта

for i in range(2\*n+1):

for j in range(2\*n+1):

if j % 2 == 0:

maze[i][j] = '+'

if j % 2 == 1:

maze[i][j] = '-'

if i % 2 == 1 and j % 2 == 0:

maze[i][j] = '|'

# закрытие лабиринта (периметр)

for j in range(2\*n+1):

if j % 2 == 0:

maze[0][j] = "+"

maze[2\*n][j] = "+"

else:

maze[0][j] = "-"

maze[2\*n][j] = "-"

for i in range(2\*n+1):

if i % 2 == 0:

maze[i][0] = "+"

maze[i][2\*n] = "+"

else:

maze[i][0] = "|"

maze[i][2\*n] = "|"

# выбор случайной комнаты и пометка ее как посещенной

visited = []

current\_cell = (random.randint(0, n-1)\*2+1, random.randint(0, n-1)\*2+1)

visited.append(current\_cell)

maze[current\_cell[0]][current\_cell[1]] = '.'

# генерация лабиринта с помощью алгоритма Recursive backtracker

while len(visited) < n\*n:

neighbors = []

if current\_cell[0] > 1 and (current\_cell[0]-2, current\_cell[1]) not in visited:

neighbors.append((current\_cell[0]-2, current\_cell[1]))

if current\_cell[0] < 2\*n-1 and (current\_cell[0]+2, current\_cell[1]) not in visited:

neighbors.append((current\_cell[0]+2, current\_cell[1]))

if current\_cell[1] > 1 and (current\_cell[0], current\_cell[1]-2) not in visited:

neighbors.append((current\_cell[0], current\_cell[1]-2))

if current\_cell[1] < 2\*n-1 and (current\_cell[0], current\_cell[1]+2) not in visited:

neighbors.append((current\_cell[0], current\_cell[1]+2))

if len(neighbors) > 0:

next\_cell = random.choice(neighbors)

visited.append(next\_cell)

maze[next\_cell[0]][next\_cell[1]] = '.'

if next\_cell[0] > current\_cell[0]:

maze[current\_cell[0]+1][current\_cell[1]] = '.'

elif next\_cell[0] < current\_cell[0]:

maze[current\_cell[0]-1][current\_cell[1]] = '.'

elif next\_cell[1] > current\_cell[1]:

maze[current\_cell[0]][current\_cell[1]+1] = '.'

else:

maze[current\_cell[0]][current\_cell[1]-1] = '.'

current\_cell = next\_cell

else:

current\_cell = visited[random.randint(0, len(visited)-1)]

# Вывод в файл

with open('maze.txt', 'w') as f:

for i in range(2\*n+1):

for j in range(2\*n+1):

f.write(maze[i][j])

f.write('\n')

Код программы и вывод лабиринта (Рисунок 5.5) с применением алгоритма Recursive backtracker, соблюдая условия формирования алгоритма.

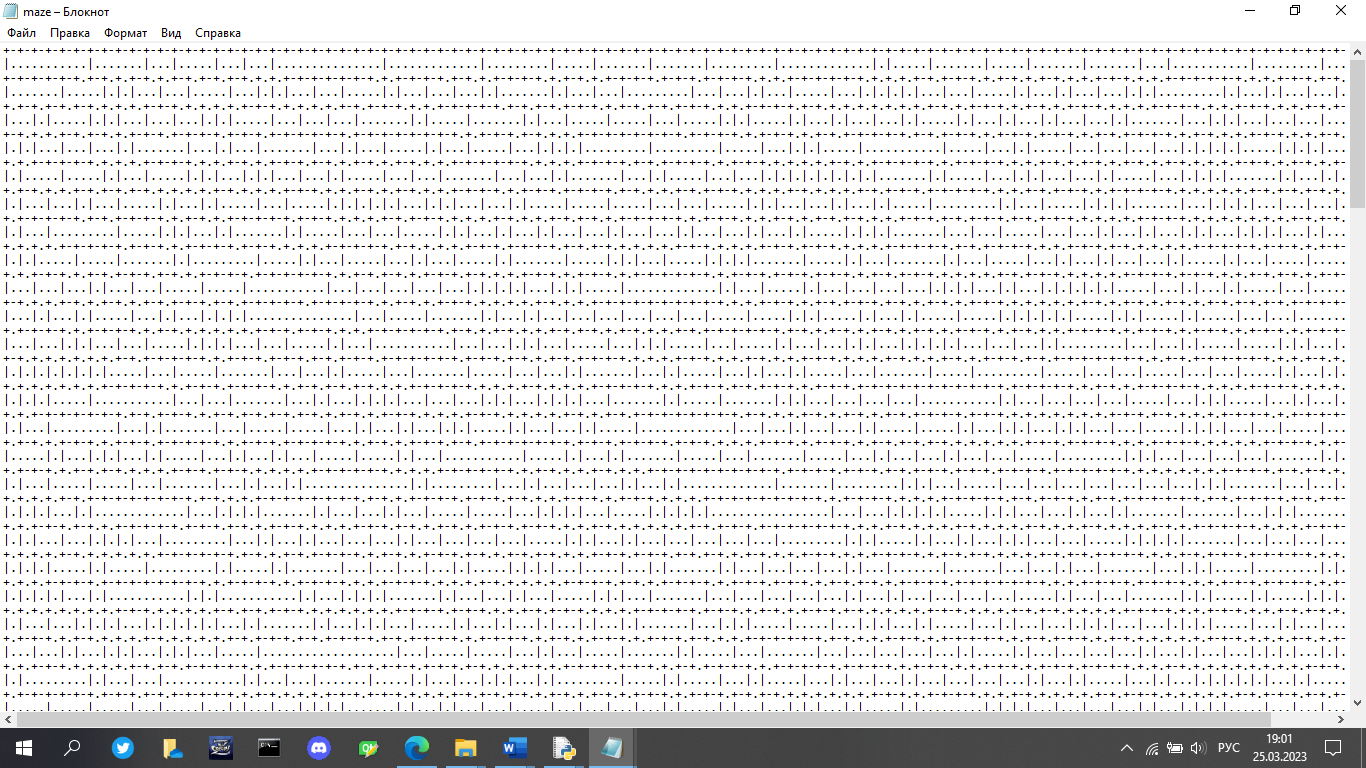


Рисунок 5.5 – Результат вывода лабиринта

**Описание программного кода:**

Данный код генерирует случайный лабиринт размером 100х100 с помощью алгоритма генерации лабиринта на основе глубокого поиска (Recursive backtracker).

В начале кода импортируется модуль random, который используется для генерации случайных чисел.

Затем задается размер лабиринта n равный 100, и создается пустой двумерный массив maze размером (2\*n+1) x (2\*n+1).

Далее лабиринт заполняется с помощью двойного цикла for. В каждой ячейке лабиринта устанавливается один из следующих символов: "+", "-", "|". Эти символы будут использоваться для отображения стен и перекрестков в лабиринте. Если j равно четному числу, то символ "+" используется для отображения перекрестка, а если j равно нечетному числу, то символ "-" используется для отображения стены. Если i равно нечетному числу, а j равно четному числу, то символ "|" используется для отображения стены.

Затем закрываются все внешние границы лабиринта. Первый и последний столбцы заполняются символами "+" или "-", а первая и последняя строки заполняются символами "|". Это создает замкнутый периметр лабиринта.

Далее выбирается случайная комната в лабиринте и помечается как посещенная. Координаты этой комнаты сохраняются в переменной current\_cell. Комната отмечается символом "." в массиве maze.

Каждая ячейка лабиринта имеет две координаты – строку и столбец. Текущая ячейка, в которой мы находимся, хранится в переменной current\_cell.

Далее идет блок if-ов, который проверяет наличие соседних ячеек и добавляет их в список соседей, если они еще не были посещены.

* Первый блок if проверяет, есть ли сверху текущей ячейки ячейка, которая еще не была посещена. Если есть, то она добавляется в список соседей. (current\_cell[0]-2, current\_cell[1]) – это координаты ячейки сверху от текущей ячейки.
* Второй блок if проверяет, есть ли снизу текущей ячейки ячейка, которая еще не была посещена. Если есть, то она добавляется в список соседей. (current\_cell[0]+2, current\_cell[1]) – это координаты ячейки снизу от текущей ячейки.
* Третий блок if проверяет, есть ли слева от текущей ячейки ячейка, которая еще не была посещена. Если есть, то она добавляется в список соседей. (current\_cell[0], current\_cell[1]-2) – это координаты ячейки слева от текущей ячейки.
* Четвертый блок if проверяет, есть ли справа от текущей ячейки ячейка, которая еще не была посещена. Если есть, то она добавляется в список соседей. (current\_cell[0], current\_cell[1]+2) – это координаты ячейки справа от текущей ячейки.

Если список соседей не пустой, то мы выбираем случайную ячейку из списка соседей, отмечаем ее как посещенную, соединяем ее с текущей ячейкой (удаляем стену между ними), делаем выбранную ячейку текущей и повторяем процесс.

Если список соседей пустой, то мы возвращаемся на предыдущую ячейку и продолжаем поиск соседей от нее.

Этот процесс продолжается до тех пор, пока мы не посетим все ячейки лабиринта.

Первая строка "with open ('maze.txt', 'w') as f:" открывает файл с именем "maze.txt" в режиме записи ('w'), а также создает временный объект 'f', который будет использоваться для записи в файл.

Затем следует два цикла for, которые заполняют содержимое файла. Первый цикл for выполняется 2n+1 раз, а второй цикл for также выполняется *2*n+1 раз для каждой итерации первого цикла.

Внутри вложенного цикла for происходит запись содержимого лабиринта в файл. Каждый элемент лабиринта записывается в файл с помощью метода write объекта 'f'.

После записи каждой строки в файл, метод write ('\n') вызывается для добавления символа переноса строки, чтобы перейти на следующую строку в файле.

Наконец, блок 'with' закрывает файл, когда запись завершена, что гарантирует, что файл будет сохранен и освобождает ресурсы, занимаемые временным объектом 'f'.

# **ЗАКЛЮЧЕНИЕ**

В результате данной учебной практики были сделаны выводы о том, что регулярная практика необходима для закрепления навыков. Так же были усовершенствованы навыки работы со стандартными приложениями такими как: Блокнот, Microsoft Visual, Paint 3D. На практике были проявлены следующие умения: решать задачи за короткий период времени, быстро ориентироваться в условиях задачи и находить наиболее оптимальные решения.

Таким образом, программа учебной практики ПМ.02 в период с 01.09.2022 по 23.12.2022 была выполнена в полном объеме.

При выполнении работ, предусмотренных заданием на практику, в организации ФГБОУ ВО «Вятский государственный университет» мной, Червяковой Натальей Александровной, соблюдались правила внутреннего трудового распорядка, охраны труда, техники безопасности, противопожарной безопасности, санитарно-эпидемиологические правила и гигиенические нормативы. Пройден вводный инструктаж и инструктаж на рабочем месте. Замечаний не имела.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Fun with Python #2: Rat in a Maze [Электронный ресурс]. **–**(<https://ozekai9.medium.com/fun-with-python-2-rat-in-amazeaa74d5c7b2dc>).
2. Александра Иванова – Программирование на Pascal [Электронный ресурс]. – (<https://stepik.org/course/3352/syllabus>).
3. Головоломки Пятнашки. – URL: <https://crossword.nalench.com/golovolomki-fifteen.html> (дата обращения: 10.10.2022)
4. Долинер, Л. И. Основы программирования в среде PascalABC.NET: учебное пособие /Л.И. Долинер. – Екатеринбург: Изд-во Урал. ун-та, 2014. – 128 с.
5. Лабиринты: классификация, генерирование, поиск решений [Электронный ресурс]. **–** (<https://temofeev.ru/info/articles/labirinty-klassifikatsiya-generirovanie-poiskresheniy/?ysclid=lcx7nwthwq26545301/>).
6. Любанович Билл. Простой Python. Современный стиль программирования. — СПб.: Питер, 2016. — 480 с.: ил. — (Серия «Бестселлеры O’Reilly»).
7. Основы программирования на языке Pascal [Электронный ресурс]. – (<https://stepik.org/course/60717/syllabus>).
8. Прохоренок, Н. А. Python 3 и PyQt 5. Разработка приложений. — 2-е изд., перераб. и доп. /Н. А. Прохоренок, В. А. Дронов. — СПб.: БХВ-Петербург, 2018. — 832 с.: ил. — (Профессиональное программирование).
9. Свейгарт Эл**.** Python. Чистый код для продолжающих. — СПб.: Питер, 2022. — 384 с.: ил. — (Серия «Библиотека программиста»).