Оглавление

[ВВЕДЕНИЕ 2](#_Toc70541839)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 3](#_Toc70541840)

[1.1 Описание задачи о назначениях 3](#_Toc70541841)

[1.2 Подходы к решению задачи о назначениях 3](#_Toc70541842)

[1.3 Венгерский алгоритм для решения задачи о назначениях 3](#_Toc70541843)

[2 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ВЕНГЕРСКОГО АЛГОРИТМА 4](#_Toc70541844)

[2.1 Подготовка и настройка среды разработки 4](#_Toc70541845)

[2.2 Разработка методов для работы с матрицами на PHP 4](#_Toc70541846)

[2.3 Разработка пользовательского интерфейса 4](#_Toc70541847)

[2.4 Пошаговая программная реализация венгерского алгоритма 4](#_Toc70541848)

[2.5 Тестирование программы и описание контрольного примера реализации 4](#_Toc70541849)

[ЗАКЛЮЧЕНИЕ 5](#_Toc70541850)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 6](#_Toc70541851)

# ВВЕДЕНИЕ

В настоящее время очень активные обороты набирает автоматизация различных вычислений и процессов в социально-экономической сфере. Одним из главных назначений автоматизации в данной сфере является принятие различных решений. Учитывая вышесказанное становится актуальным применений автоматизированных систем поддержки принятия решений.

Одной из наиболее популярных и полезных задач в сфере принятия решений является «задача о назначениях», которая позволяет определить оптимальное распределение исполнителей по задачам, которые необходимо выполнить. По сути, задача представляет из себя матрицу значений. Эти значения – показатель заработной платы, которую необходимо выплатить исполнителю данной задачи (сверху вниз – исполнитель, слева направо - задачи). Помимо показателя заработной платы, который надо привести к наименьшему значению, также может быть показатель прибыли, который надо максимизировать. В рамках данной курсовой работы будет решена задача, которую надо привести к минимальному значению – распределение исполнителей заявок на ремонт оборудования, где значения матрицы – время выполнения исполнителей данного типа работ.

Целью данной курсовой работы является разработка программы для решения задачи о назначениях. Для достижения цели необходимо решить следующие задачи:

1. Изучить алгоритмы, которые используются для решения задачи о назначениях
2. Выбрать любой алгоритм для решения задачи
3. Выбрать среду разработки программы для решения задачи
4. Подготовить и настроить среду разработки
5. Разработать методы для работы с матрицами. Разработать программу и протестировать её, сделав контрольный пример реализации системы.

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## Описание задачи о назначениях

Задача о назначениях является одной из самых популярных и часто решаемых задач в сфере принятия решений. Суть задачи о назначениях заключается в том, что необходимо найти оптимальный план выполнения работ таким образом, чтобы затраты на выплату заработной платы были минимальные. Вся задача представляется в виде матрицы, обязательно, квадратной, так как на одну задачу не может быть назначено более одного исполнителя, а один исполнитель может быть занят только одной задачей. Исходя из этого, следует отметить, что при решении данной задачи количество исполнителей и выполняемых задач должно быть равным.

## Венгерский алгоритм для решения задачи о назначениях

Наиболее распространённым способом решения задачи о назначениях является венгерский алгоритм. Венгерский алгоритм состоит из нескольких шагов, и позволяет получить n значений в разных столбцах, которые будут показывать какую задачу будет решать исполнитель. В данном случае, n – размер матрицы. Рассмотрим шаги венгерского алгоритма более подробно на примере матрицы, приведённой на рисунке 1.

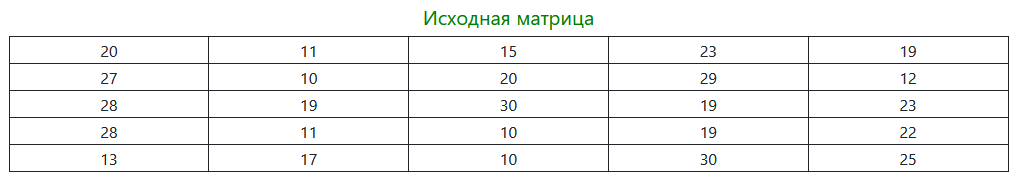


Рисунок 1 – Пример матрицы для решения задачи о назначениях

Считаем, что исполнители задач расположены слева сверху вниз, а наименования задач сверху слева направо. В таком случае, Исполнитель 1 за выполнение задачи 1 требует 20 условных денежных единиц, за задачу 2 – 11 условных денежных единиц и так далее. Следовательно, любой элемент матрицы – цена выполнения работы в этом столбце исполнителем по этой строке.

На первом шаге алгоритма необходимо провести так называемую, редукцию матрицы по строкам. Суть редукции заключается в том, что необходимо выполнить поиск минимального в строке элемента, а затем этот элемент вычесть из каждого элемента этой строки. На рисунке 2 показан результат редукции матрицы по строкам. Жёлтым цветом выделены минимальные значения из исходной матрицы. Эти значения нулевые, так как в процессе преобразования они вычитались из себя.



Рисунок 2 – матрица после редукции по строкам

Далее, венгерский алгоритм предполагает проведение той же операции, что и на первом шаге, только с перебором по столбцам. Основная идея этого шага в том, что мы в любом случае получим нулевые значения по столбцам, в которых их нет. На рисунке 3 можно увидеть результат редукции матрицы по столбцам. От матрицы, показанной на рисунке 2 она отличается тем, что имеет нулевые значения в столбцах 0 и 4.

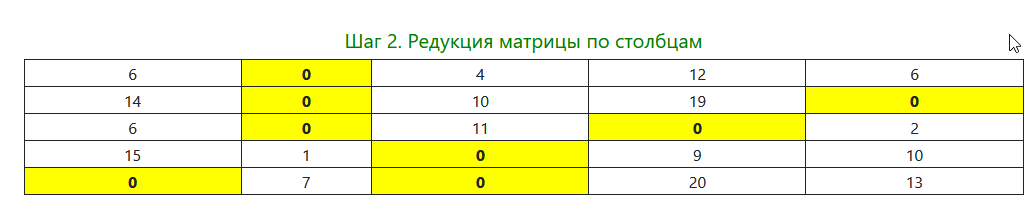


Рисунок 3 – матрица после редукции по столбцам

Следующим шагом венгерского алгоритма является вычёркивание строк и столбцов с нулевыми значениями, таким образом, чтобы осталось несколько свободных элементов, с которым в дальнейшем можно работать. Вычёркиваний должно быть минимально возможное количество. Для того, чтобы провести вычёркивание элементов правильно, стоит удалить наибольшее количество нулей, затем ориентироваться по ситуации. Если в строке или столбце есть только один ноль, то вычеркнуть можно либо строку, либо столбец на своё усмотрение. На рисунке 4 предложен способ вычёркивания лишних элементов из матрицы, расположенной на рисунке 3. Элементы, которые не закрашены тёмным цветом, можно использовать на дальнейших шагах алгоритма.

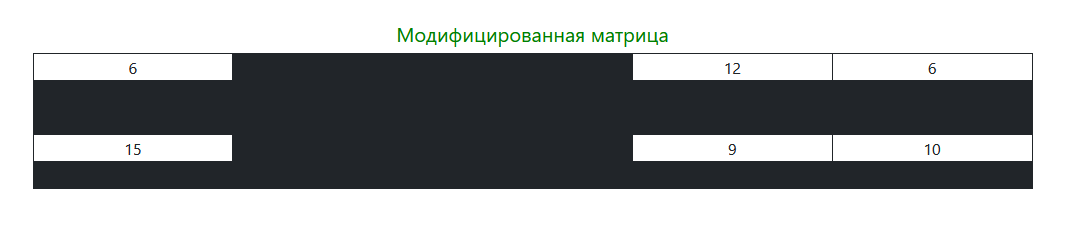


Рисунок 4 – преобразованная матрица

Далее необходимо выбрать минимальное значение из тех, которые видны. Посредством несложных вычислений понятно, что в данном случае этот элемент – 6. Далее, необходимо провести операции по изменению вычеркнутых элементов. Логика операций такая – если элемент вычеркнут единожды, то его надо оставить без изменений, дважды – минимальный элемент надо прибавить к этому значению, а от свободных элементов отнять минимальное значение. Данная операция показана на рисунке 5.



Рисунок 5 – оптимальный план распределения исполнителей по задачам

Рисунок 5 – матрица после преобразования по венгерскому алгоритму

Следующий шаг – сопоставление полученной матрицы (рисунок 5) с исходной матрицей. Необходимо взять нулевое значение из каждого столбца и сопоставить его со значением по тому же адресу в исходной матрице. Это значение и есть показатель из оптимального плана распределения работ. Очень важно, чтобы выбранное значение было единственным в строке или столбце, по причине того, что исполнитель не может быть занят выполнением двух задач, а одна задача не может выполняться двумя исполнителями.

Обратимся к рисунку 1 для сопоставления. Для простоты расчётов идём справа налево по столбцам матрицы. Рассмотрим нули, выделенные жёлтым в первой нулевой и первой строках. Нулевая строка не подходит, так как в этой же строке первого столбца есть значение, представленное единственным в данном столбце. Если возьмём значение из нулевой строки в последней задаче, то задача №1 останется без исполнителя. Следовательно, берём первую строку и получаем, что задачу 4 будет выполнить исполнитель 1 за 12 денежных единиц. В столбце 3 берём единственное значение на нулевой 2 строке, так как оно уникальное для данного пересечения строки и столбца. Здесь не требуется сопоставления с другими столбцами. Та же ситуация складывается во втором столбце матрицы. Выходит, что для 3 и 2 задачи подходят исполнители 2 и 3 соответственно. Это стоит 19 и 10 денежных единиц. Далее настаёт очередь первого столбца, в котором мы выбираем единственный возможный 0, который есть в данном столбце. Это стоит 11 денежных единиц. В последнем столбце выбираем нижнее нулевое значение. Верхнее взять не можем, так как его исполнитель уже занят на задаче №1.

Таким образом, получили 5 значений, которые символизируют количество заработной платы для выполнения задач. Результат:

1. Задача № 0 – исполнитель №4 (13 единиц)
2. Задача №1 – исполнитель №0 (11 единиц)
3. Задача №2 – исполнитель №3 (10 единиц)
4. Задача №3 – исполнитель №2 (19 единиц)
5. Задача №4 – исполнитель №1 (12 единиц)

Таким образом, получили производственный план, следуя которому расходы на ЗП будут наименьшими, а также каждый исполнитель занят только 1 делом.

# ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ВЕНГЕРСКОГО АЛГОРИТМА

## Подготовка и настройка среды разработки

Разработка программы для решения задачи о назначениях осуществлена на языке программирования высокого уровня PHP. Для работы на данном языке необходима хорошая среда разработки, локальный сервер, который позволит запускать и исполнять скрипты прямо в браузере. Данный язык программирования выбран для разработки, так как он позволяет создавать элементы вёрстки, взаимодействуя с языком HTML. На основе этих возможностей разработан практически весь интерфейс программы, также были разработаны скрытые методы с интерфейсом, которые используются для отладки и тестирования работы кода.

Что касается редактора кода – для разработки используется программа PHPStorm. Данная среда разработки позволяет работать с кодом и предоставляет множество полезного функционала, например, подсветку синтаксиса, анализ кода, соединение с сервером хостинг-провайдера, соединение с GIT для контроля версий кода и командной разработки. Данная программа является платной, стоимость базовой лицензии составляет около 20 000 рублей в год, но JetBrains позволяет приобретать бесплатные лицензии для студентов и преподавателей, а также сотрудников высших учебных заведений.

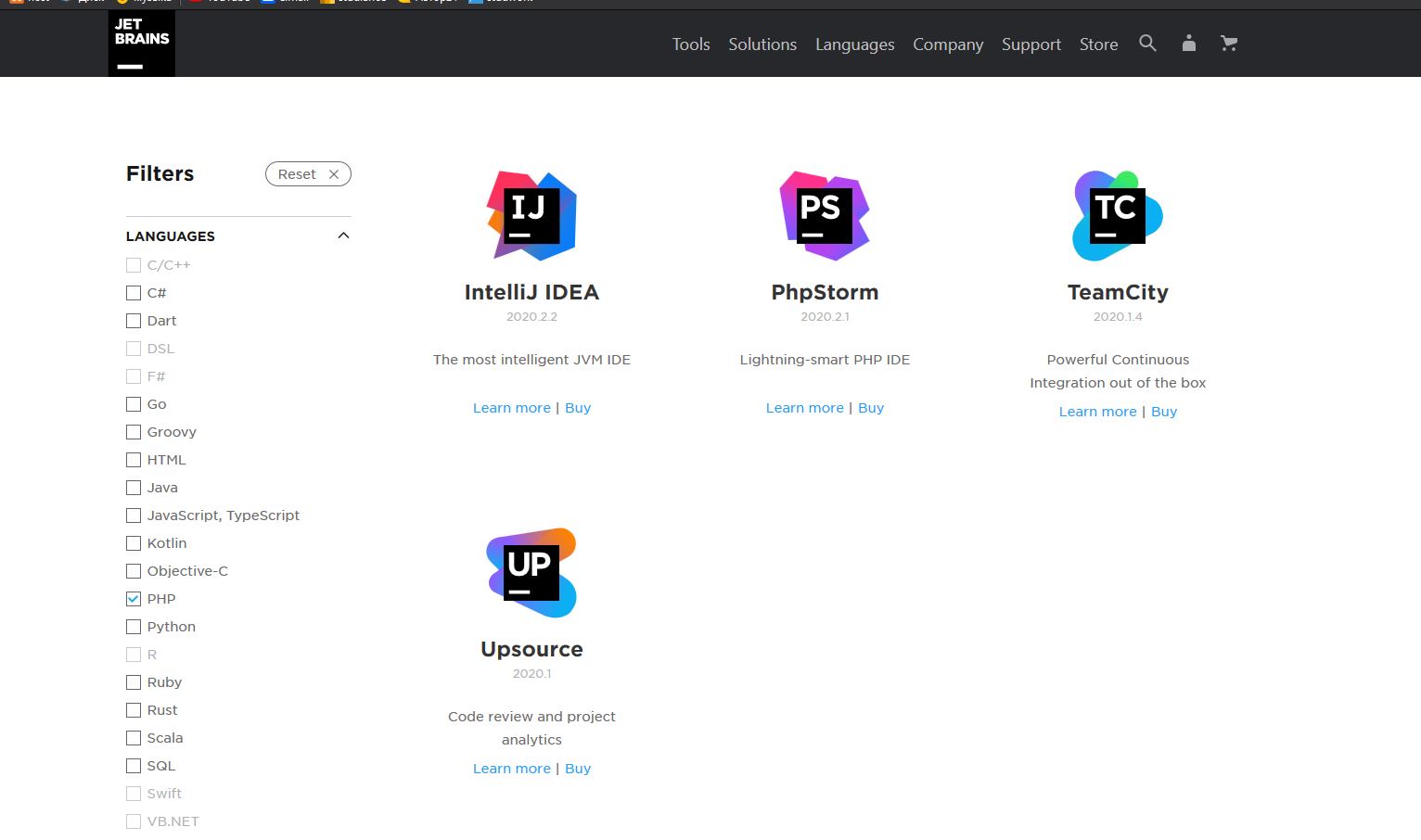


Рисунок 4 – скачивание программы PHPstorm

Для приобретения студенческой лицензии данной программы необходимо привязывать к аккаунту ту почту, которая может подтвердить принадлежность к той или иной категории пользователей. Рассмотрим процесс установки среды разработки и создания проекта. На сайте компании JetBrains пользователям предлагается выбрать язык программирования, а скрипт сам подберёт ПО для работы с этим языком. Например, выбрав PHP, мы видим, что на выбор предложено 4 разные программы, одна из которых PHPStorm. При регистрации на студенческую почту, лицензию программы можно получить совершенно бесплатно на 1 год.

PHPStorm является полностью самостоятельной средой разработки, которая позволяет программисту очень многое. Предусмотрена как самостоятельная, так и командная разработка ПО, есть возможность подключиться к системе контроля версий GIT и другим. Одна из важнейших способностей программы – настройка прямого FTP доступа к сайту при наличии пароля. В таком случае сайт автоматически выгрузится на локальный сервер, можно вносить в него изменения, не боясь «сломать» основной сайт.

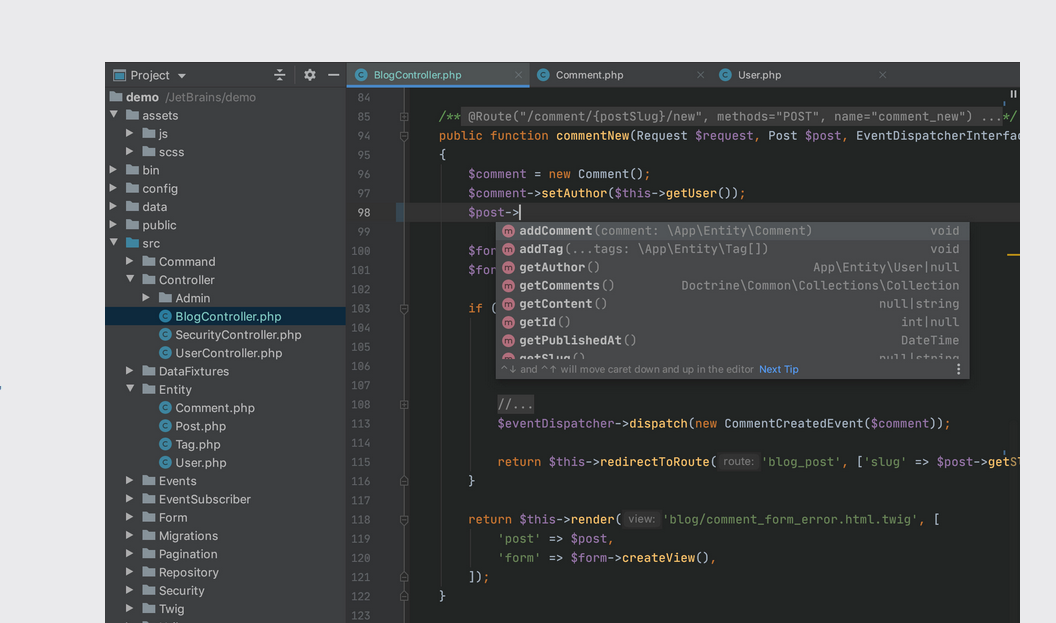


Рисунок 5 – интерфейс программы PHPstorm

## Разработка методов для работы с матрицами на PHP

Для работы с матрицами при разработке программы было разработано несколько самописных функций для быстрой работы с матрицами. К таким функциям стоит отнести транспонирование матрицы, функция исключения строк и столбцов, неугодных венгерскому алгоритму, функции проверки валидности строки и столбца в матрице, вывод матрицы на экран в виде таблицы и другие. Рассмотрим более подробно каждую функцию.

Транспонирование матрицы. Представляет из себя «переворачивание» матрицы посредством записи текущей строки в столбец. Данная функция используется несколько раз в работе других самописных функций.

Печать матрицы. Функция принимает матрицу в качестве первого аргумента, вторым аргументом является название матрицы. Результаты работы функции представлен на рисунке 6 (строка 24 данного кода). Особенность функции – все нули выделяются жёлтым для удобства пользовательского интерфейса.

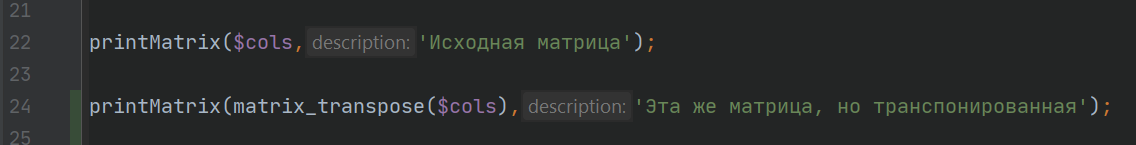




Рисунок 6 – транспонирование матрицы.

Редукция матриц по строкам и столбцам. Представляет из себя простой перебор матрицы по элементам, поиск минимума по каждой строке / столбцу. Минимумы вычитаются из каждого элемента этой строки / столбца. Для быстрого осуществления этих операций созданы функции под названием ReduxMatrixByRows(матрица) и ReduxMatrixByCols(матрица). Результаты работы этих функций представлены на рисунке 7 (сопоставлять необходимо с верхней матрицей на рисунке 6).



Рисунок 7 – редукции матриц по строкам

Исключение строк и столбцов из матрицы представляет из себя достаточно сложный алгоритм. Вначале перебирается вся матрица в двойном цикле, создаётся пустой объект, в котором есть два свойства – номер стоки, в котором расположен 0, и номер его столбца. Данные этого объекта пишутся в массив, затем объект удаляется и перезаписывается заново для следующих нулей. В результате, на данном шаге алгоритма получаем массив объектов с адресами нулей. Также в каждом объекте предусмотрено хранение информации о количестве нулей в столбце и строке для каждого отдельно взятого нуля. Это пригодится при использовании массива объектов для перечёркивания строк и столбцов матрицы. Проблема данного массива объектов в том, что из-за специфики кода некоторые объекты в нём продублированы. Задача заключается в том, чтобы получить массив объектов с уникальными значениями, а также чтобы индекс этого массива начинался с нуля и имел строгий порядок увеличения на единицу. Для решения этой задачи необходимо применить несколько базовых PHP-функций для работы с массивами. После решения проблемы получили массив объектов, который по сути, представляет из себя рекомендации от программы, какие строки и столбцы необходимо вычеркнуть. Далее в функции срабатывают два цикла (последовательно) в котором идёт перебор полученного массива объектов в функции IsRowValid и isColValid, которые возвращают логическое значение в зависимости от пригодности строки или столбца. Отдельный цикл работает для строк, отдельный для столбцов матрицы. Если для текущего элемента найден 0, то идёт проверка на валидность строки и столбца. Если выясняется, что строка или столбец не валидны, то к значению элемента присоединяется строка null (типа string). Данный подход полезен при проверке статуса закрытости элемента (1 – если элемент вычеркнут единожды, 2 – дважды, 0 – элемент свободен вычёркивания). Если на текущем нуле будет найдено значение null – строка вычеркнута единожды, nullnull – дважды. Иначе элемент свободен. Это в дальнейшем поможет при обработке плана затрат. Результат работы данной функции представлен на рисунке ниже (данный вывод матрицы не участвует в конечном интерфейсе, создавался исключительно для отладки кода)

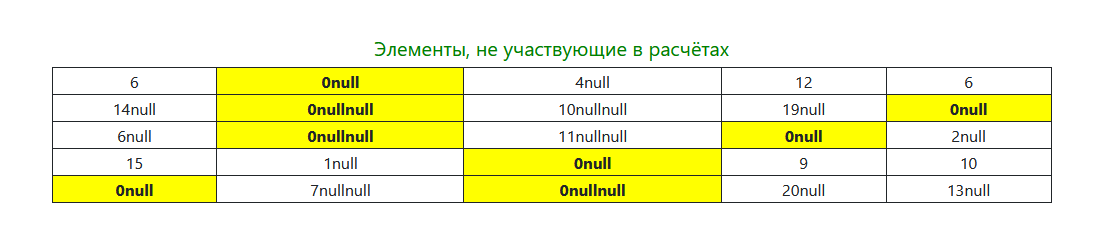
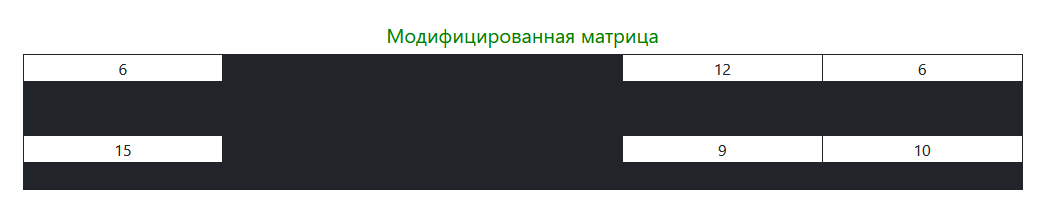


Рисунок 8 – матрица лишних элементов

Далее создаём вычеркнутую матрицу (она уже для пользователя). Передаём туда вышеописанную матрицу, вычеркиваем тёмным цветом ненужные элементы и выводим таблицу на экран. Результат представлен на рисунке 9. Сопоставлять необходимо с матрицей на рисунке 9.



Далее, матрица, созданная в функции, описанной выше, передаётся в следующую функцию, которая перебирает эту матрицу, и на основе её создаёт подматрицу, которая имеет три типа элементов: 0 – свободный элемент, 1 – вычеркнутый однократно, 2 – вычеркнутый дважды. Важно отметить, что размер новой матрицы всегда рамен размеру исходной матрицы. Далее, запускается следующий цикл, который использует матрицу, полученную после редукции по столбцам и изменяет её элементы исходя из правил венгерского алгоритма. Свободные элементы уменьшаются на размер минимального значения из массива элементов, которые не были вычеркнуты при преобразовании, вычеркнутые однократно игнорируются, к дважды вычеркнутым прибавляется минимальное значение. Таким образом, на выходе из этой функции получаем функцию, которая создаёт матрицу, в которой жёлтым цветом выделены нули, указывающие, какие элементы (необходимо сопоставлять адреса) являются пересечением исполнителя и задачи. Из этих элементов складывается оптимальный план решения задачи.

## Разработка пользовательского интерфейса

Для наибольшей универсальности задачи предусмотрен ввод размерности матрицы для решения задачи. Интерфейс ввода размерности матрицы представлен на рисунке 9.

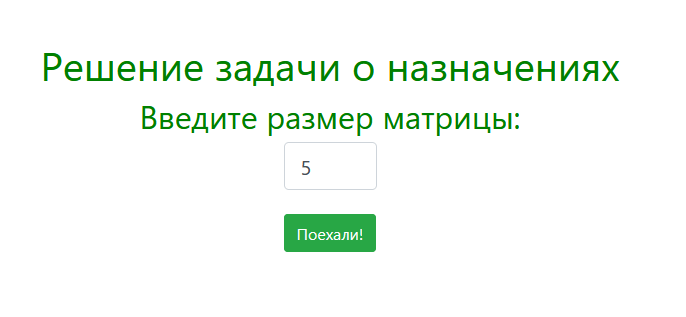
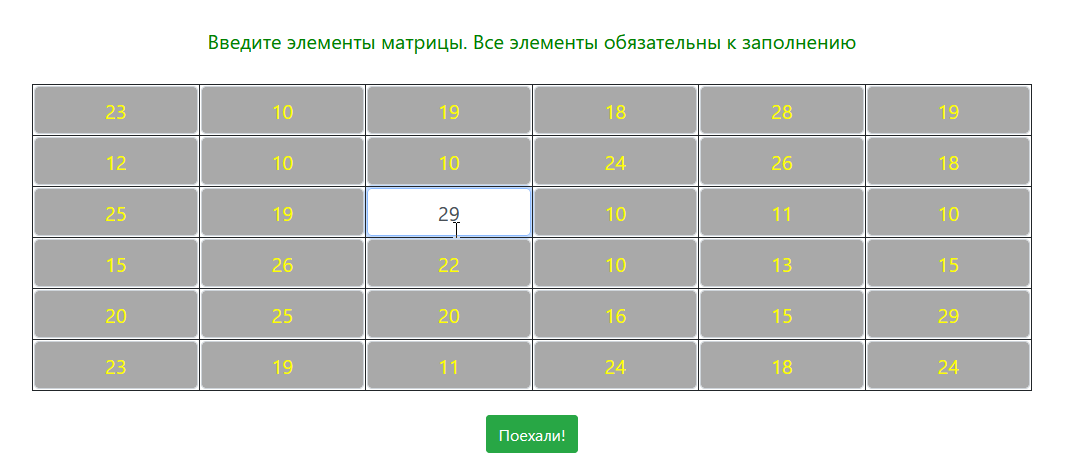


Рисунок 9 – ввод размера матрицы

Далее пользователю предлагается осуществить ввод матрицы. Интерфейс ввода матрицы представлен на рисунке 10.



Если будут найдены пустые поля, все отправка матрицы на решение задачи будет невозможна. После отправки матрицы на расчёты, пользователь увидит результат редукции матрицы

## Тестирование программы и описание контрольного примера реализации

# ЗАКЛЮЧЕНИЕ

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ