**АННОТАЦИЯ**

Данная выпускная квалификационная работа описывает современный метод оптимизации решения задачи теории расписаний. Целью данной работы является разработка и исследование иерархической системы и метода построения комплексного расписания групповой обработки партий требований.

Ключевые слова: иерархическая система, конвейерная обработка, оптимизация, требования, партии, группы, расписание.

В работе представлена иерархическая модель построения расписания групповой обработки данных различных типов при условии на временное ограничение работы всей системы, методы построения партий данный фиксированного и всех типов и метод построения и оптимизации расписания.

На основе полученных методов был разработан модульный программный продукт, описанный в разделе 4.

Для доказательства работоспособности этой системы в целом в разделе 5 приведен сравнительный анализ всей системы с работой системы без использования модуля оптимизации составов партий данных.

Экономическое обоснование программного продукта приведено в разделе 6.

Показатели охраны труда и методы их улучшение приведены в разделе 7.

В представленной пояснительной записке к выпускной квалификационной работе 7 основных разделов на 123 страницах, включающих:

* 4 таблицы;
* 10 рисунков (включая 2 структурные схемы).

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 6](#_Toc453803732)

[1 ОБОСНОВАНИЕ АКТУАЛЬНОСТИ РАЗРАБОТКИ МЕТОДОВ ОПТИМИЗАЦИИ СОСТАВОВ ПАРТИЙ ДАННЫХ 11](#_Toc453803733)

[1.1 Постановка задачи 11](#_Toc453803734)

[1.2 Анализ существующих методов построения комплексных расписаний обработки партий данных 13](#_Toc453803735)

[1.3 Обоснование подхода к решению задачи построения расписаний обработки партий данных разных типов при наличии ограничения на интервал функционирования системы 15](#_Toc453803736)

[1.4 Обоснование использования аппарата теории иерархических и неантагонистических игр при построении расписаний групповой обработки партий данных 17](#_Toc453803737)

[Выводы раздела 1 18](#_Toc453803738)

[2 МОДЕЛЬ СИСТЕМЫ ПОСТРОЕНИЯ РЕШЕНИЯ ПО СОСТАВАМ ПАРТИЙ ДАННЫХ 19](#_Toc453803739)

[2.1 Описание иерархической модели построения комплексного расписания обработки партий данных 19](#_Toc453803740)

[2.2 Обоснование способа формирования решения по составам партий данных заданного типа 28](#_Toc453803741)

[2.3 Метод построения решения по составам партий данных заданного типа данных 31](#_Toc453803742)

[2.4 Описание метода формирования решения по составам партий заданного типа данных 32](#_Toc453803743)

[2.5 Метод построения решения по составам партий различных типов данных 38](#_Toc453803744)

[2.6 Обоснование метода построения решения по составам партий различных типов данных 38](#_Toc453803745)

[2.7 Описание метода формирования составов партий различных типов данных 40](#_Toc453803746)

[2.8 Метод формирования составов партий различных типов данных 40](#_Toc453803747)

[2.9 Анализ эффективности метода построения решения по составам партий различных типов данных 46](#_Toc453803748)

[Выводы раздела 2 46](#_Toc453803749)

[3 МЕТОД ОПРЕДЕЛЕНИЯ ПОРЯДКА ОБРАБОТКИ ПАРТИЙ ДАННЫХ (РАСПИСАНИЙ ОБРАБОТКИ ПАРТИЙ ДАННЫХ) 47](#_Toc453803750)

[3.1 Обоснование метода определения порядка обработки партий данных заданных типов 47](#_Toc453803751)

[3.2 Описание алгоритма формирования эффективного расписания групповой обработки партий данных 49](#_Toc453803752)

[Выводы раздела 3 53](#_Toc453803753)

[4 ОПИСАНИЕ РАЗРАБОТАННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 54](#_Toc453803754)

[4.1 Описание разработанного модуля построения решения о составе партий данных фиксированного типа 54](#_Toc453803755)

[4.2 Описание разработанного модуля построения решения о составах партий данных рассматриваемых типов 57](#_Toc453803756)

[Выводы раздела 4 60](#_Toc453803757)

[5 АНАЛИЗ РЕЗУЛЬТАТОВ РАБОТЫ МЕТОДОВ ПОСТРОЕНИЯ РЕШЕНИЯ ПО ОБРАБОТКЕ ПАРТИЙ ДАННЫХ 61](#_Toc453803758)

[Выводы раздела 5 65](#_Toc453803759)

[6 ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА 66](#_Toc453803760)

[6.1 Маркетинговые исследования программного продукта 66](#_Toc453803761)

[6.1.1 Сущность и этапы маркетинговых исследований 66](#_Toc453803762)

[6.1.2 Оценка рыночной направленности программного продукта 66](#_Toc453803763)

[6.1.3 Конкурентоспособность продукта 67](#_Toc453803764)

[6.1.4 Сегментация рынка 67](#_Toc453803765)

[6.1.5 Предполагаемый потребитель 68](#_Toc453803766)

[6.1.6 Жизненный цикл программного продукта 68](#_Toc453803767)

[6.1.7 Уровни программного продукта как товара 68](#_Toc453803768)

[6.1.8 Итоги маркетингового исследования 69](#_Toc453803769)

[6.2 Определение затрат на проектирование продукта 69](#_Toc453803770)

[6.2.1 Расчет трудоемкости программного продукта 69](#_Toc453803771)

[6.2.2 Расчет эксплуатационных затрат разработчика 70](#_Toc453803772)

[6.2.3 Расчет сметы затрат на проектирование 71](#_Toc453803773)

[6.3 Расчет капитальных затрат 71](#_Toc453803774)

[6.4 Формирование цены предложения разработчика 72](#_Toc453803775)

[6.5 Оценка экономической эффективности программного продукта 73](#_Toc453803776)

[Выводы раздела 6 74](#_Toc453803777)

[7 БЕЗОПАСНОСТЬ ЖИЗНИДЕЯТЕЛЬНОСТИ 75](#_Toc453803778)

[7.1 Основные сведения 75](#_Toc453803779)

[7.2 Краткая характеристика помещения и выполняемых работ 76](#_Toc453803780)

[7.3 Планировка и размещение оборудования и рабочих мест 77](#_Toc453803781)

[7.4 Тяжесть и напряженность труда. режим труда и отдыха 78](#_Toc453803782)

[7.5 Качество воздуха и микроклимат рабочей зоны 78](#_Toc453803783)

[7.6 Шум и вибрация 79](#_Toc453803784)

[7.7 Освещение 79](#_Toc453803785)

[7.8 Электро - и пожаробезопасность 80](#_Toc453803786)

[7.9 Статическое электричество и излучения 81](#_Toc453803787)

[7.10 Эргономика и техническая эстетика 82](#_Toc453803788)

[7.11 Расчет искусственного освещения производственных помещений 82](#_Toc453803789)

[Выводы раздела 7 84](#_Toc453803790)

[ЗАКЛЮЧЕНИЕ 85](#_Toc453803791)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 88](#_Toc453803792)

[ПРИЛОЖЕНИЕ А 90](#_Toc453803793)

[ПРИЛОЖЕНИЕ Б 106](#_Toc453803794)

[ПРИЛОЖЕНИЕ В 113](#_Toc453803795)

[ПРИЛОЖЕНИЕ Г 117](#_Toc453803796)

# ВВЕДЕНИЕ

Актуальность темы. На современном этапе развития направления теории расписаний и задач оптимизации рассматриваются как приближенные, так и точные методы. В информационных технологиях часто появляется необходимость обработки больших массивов разнотипных данных за ограниченное время. Одним из способов решения данной проблемы является применение систем конвейерного типа. В этих системах важной составляющей является поступление требований на обработку. Теория расписаний является частью исследования операций. Теория расписаний исследует задачи, в которых необходимо упорядочить или, другими словами, определить последовательность выполнения совокупности работ, использования каких-либо средств и т.д.

Задачи упорядочения носят самый общий характер. Они возникают там, где существует возможность выбора той или иной очередности выполнения работ: при распределении работ на производстве, составлении расписания приземления самолетов, составлении расписания движения поездов, обслуживании клиентов в обслуживающих системах и т.д.

Результаты, к которым приводит то или иное упорядочение, существенно отличаются. В ряде практических случаев эти различия принимают стоимостный характер или определяются какой-либо другой величиной.

Цель и задачи работы. Целью выпускной квалификационной работы является разработка методов построения решения по оптимизации составов партий данных и расписаний их обработки. Для решения поставленной задачи была введена трехуровневая иерархическая модель. На основании полученной модели были поставлены задачи:

* формирование метода оптимизации составов партий всех типов данных;
* формирование метода оптимизации составов партий фиксированного типа данных;
* формирование метода оптимизации расписаний обработки партий данных при наличии ограничения на время функционирования системы.

Предмет и объект исследования. В выпускной квалификационной работе рассматривается многоуровневая система теории расписания. Теория расписаний ‑ раздел дискретной математики, занимающийся проблемами упорядочения. В большинстве случаев в теории расписания ставится задача дискретной оптимизации: построить расписание, минимизирующее время выполнения работ, стоимость работ и т.п. Расписание ‑ указание, на каких машинах и в какое время должны обслуживаться требования (выполняться работы).

Производительность обработки данных при выполнении программ можно повысить путём конвейеризации [1]. Под конвейеризацией выполнения программы подразумевается разделение её на фрагменты, каждый из которых закреплён за соответствующим сегментом конвейера. Выполнение программ реализуется в многостадийной последовательной обрабатывающей системе с заданным порядком сегментов конвейера. Сегменты конвейера следуют строго друг за другом, местоположение сегмента в последовательности определяется его номером. Обозначим количество сегментов в конвейере через *L*, тогда порядковый номер сегмента, входящего в состав конвейера - . Если в состав конвейера входит *L* сегментов, то все выполняемые в системе программы должны быть разделены на *L* фрагментов, каждый из которых закреплён для выполнения за соответствующим сегментом конвейера. Тогда выполнение конвейеризированной программы в составе конвейера предполагает строгое закрепление её *l*-того фрагмента для выполнения на *l*-том сегменте конвейера.

В систему обработки данных могут поступать различные типы данных, обозначим их количество через *n*. Через *i* обозначим номер множества однотипных данных, которые должны быть обработаны в системе, тогда . Количество элементов в множестве однотипных данных, характеризуемых индексом *i*, обозначим через . Все данные *i*-того типа (в количестве ) обрабатываются соответствующей им программой. То есть существует  типов программ, выполняемых в системе, обрабатывающих данные *i* типа. Таким образом, в системе задаётся строгое соответствие между типом данных и обрабатывающей эти данные программой. Однократное выполнение программы *i*-го типа позволяет обработать один элемент множества данных *i*-того типа. Так как множество *i*-того типа содержит  элементов, то обрабатывающая программа *i*-того типа должна быть выполнена  число раз. Цель функционирования системы в этом случае состоит в обработке всех поступающих на её вход данных. В конвейеризированной системе, которая состоит из *L* сегментов, реализуется обработка n типов данных n типами программ. Для повышения эффективности использования ресурсов системы рекомендуется обрабатывать данные партиями [2, 3]. В этом случае «партия» - это совокупность наборов данных одного (*i*-того, ) типа, которые обрабатываются в программе непосредственно друг за другом. Группа партий – это совокупность партий, обрабатываемых в течение одного из заданных интервалов времени . При этом состав групп партий для каждого временного интервала  определяется в соответствии с вводимыми критериями эффективности.

Выполнение программ в конвейеризированной вычислительной системе требует использования её (системы) ресурсов. В первую очередь такими ресурсами являются процессорное время каждого сегмента конвейера и оперативная память, в которой хранится выполняемая программа и обрабатываемые данные. Чтобы наиболее эффективно с точки зрения использования сегментов конвейера использовать вычислительную систему необходимо планировать запуск программ различных типов, то есть составлять расписания обработки данных разных типов. Так как поступление данных *i*-того типа инициирует в конвейеризованной системе выполнение программы *i*-того типа, то под «расписанием» понимается порядок поступления данных разных типов на вход системы или, другими словами, порядок запуска программ разных типов на выполнение.

Научна новизна. В выпускной квалификационной работе рассматривается градиентный подход с использованием жадной стратегии для решения задачи о оптимизации составов партий. Градиентный метод использует понятие окрестности и шага решения. Под окрестностью понимают набор возможных решений для получения локального максимума в текущей окрестности. Под локальным максимумом понимается понятие наилучшего решения среди всех из текущей окрестности. Под шагом алгоритма понимается понятие увеличения окрестности и получения большего числа потенциальных локальных максимумов для нахождения среди них глобального оптимального (наилучшего решения для поставленной задачи).

Практическое значение работы. Рассматриваемая система применима в конвейерном производстве. Система позволяет оптимизировать составы партий данных для их обработки за фиксированное время работы. Система построена таким образом, что может использоваться в любых отраслях, применяющих конвейерную обработку данных. В текущем контексте разработанная программа доказывает необходимость использования полученной системы по сравнению с системой, основанной на фиксированных партиях (составы партий не изменяются). В дальнейшем планируется разработать программный продукт, основанный на полученной системе, конвейерной обработки спутниковых снимков земли (эта задача актуальна, так как обработка большого количества изображений занимает огромное время).

Структура работы. Выпускная квалификационная работа состоит из введения, семи основных разделов, заключения, списка использованных источников и четырех приложений.

В первом разделе обоснована актуальность разработки методов оптимизации составов партий данных.

Во втором разделе представлена модель системы и методы оптимизации составов партий данных.

В третьем разделе описан метод построения и оптимизации порядка обработки партий данных.

В четвертом разделе описана программная реализация методов, описанных в разделе два.

В пятом разделе проводится анализ результатов работы разработанных методов оптимизации.

В шестом разделе представлено экономическое обоснование проектного решения.

В седьмом разделе представлены показатели охраны труда и методы их улучшения.

В результате выполнения выпускной квалификационной работы было разработано Windows-приложение, которое строит расписания обработки данных многих типов, с учетом времени перенастройки оборудования с обработки требования одного типа на другой. Разработанная система производит оптимизацию расписания по нескольким критериям эффективности, программный комплекс условно разделён на три уровня. Каждый из них представляет собой поиск оптимума по одному или нескольким критериям.

Из-за того что обработка на всех устройствах происходит последовательно, и обработка каждой партии не может быть разбита на несколько частей, необходимо сформировать порядок поступления партий данных только на первое устройство.

# ОБОСНОВАНИЕ АКТУАЛЬНОСТИ РАЗРАБОТКИ МЕТОДОВ ОПТИМИЗАЦИИ СОСТАВОВ ПАРТИЙ ДАННЫХ

## Постановка задачи

Постановка задачи предполагает наличие ограничений на время функционирования конвейеризированной системы, обрабатывающей разные типы данных, то есть существует некоторое количество интервалов фиксированной длительности, в течение которых производится конвейеризированная обработка поступающих в систему данных. Введём обозначения: через *Z* обозначим количество интервалов обработки (тогда номер интервала ), при этом длительность интервала обозначим через .

В конвейеризированной системе, которая состоит из *L* сегментов, реализуется обработка *n* типов данных n типами программ. Для повышения эффективности использования ресурсов системы рекомендуется обрабатывать данные партиями. В этом случае «партия» - это совокупность наборов данных одного (*i*-го, ) типа, которые обрабатываются в программе непосредственно друг за другом.

Обработка данных в системе происходит в течение *Z* интервалов, длительность которых , поэтому возникает необходимость формирования групп – совокупности партий, обрабатываемых в течение одного интервала функционирования конвейеризированной системы.

Так как интервалы обработки данных строго ограничены, то проблема эффективного использования ресурсов системы ставится наиболее остро. Тогда задача составления расписаний для повышения эффективности использования ресурсов системы заключается в определении порядка обработки партий данных в каждой группе. При этом необходимо учесть, что целью работы системы является обработка максимально возможного количества данных разных типов.

Входными данными для системы построения расписаний групповой обработки данных разных типов при наличии ограничений на длительность и количество интервалов обработки являются:

* количество типов данных (*n*);
* количество элементов в множестве данных каждого типа (, );
* количество (*Z*) и длительность () интервалов функционирования системы;
* количество сегментов системы (*L*);
* длительность обработки данных *i*-того типа *l*-ым сегментом системы (фрагментом *i*-той программы);
* интервалы времени переналадки приборов с обработки данных одного типа на обработку данных другого типа.

В процессе функционирования системы необходимо определить приближенно эффективное (с точки зрения вводимых в рассмотрение критериев) количество и составы партий данных, эффективный (с точки зрения обработки максимально возможного количества данных разных типов) состав групп партий данных, а так же эффективное (с точки зрения минимального времени выполнения всей группы) расписание обработки партий данных разных типов в группе. Данная задача является сложной, поэтому требуется вертикальная декомпозиция целей, в результате которой задача будет разбита на подзадачи.

В данной работе рассматриваются только два аспекта работы системы: формирование партий и построение расписаний их обработки. Необходимо разработать метод получения оптимального количества и состава партий данных каждого типа и метод получения оптимального расписания обработки групп партий данных.

Для достижения поставленной цели необходимо решить следующие задачи:

* проанализировать существующие методы построения расписаний обработки партий данных разных типов;
* сформулировать подход к построению расписаний групповой обработки партий при наличии ограничений на длительность интервалов обработки групп;
* выбрать математический аппарат;
* построить математическую модель системы;
* разработать метод формирования эффективного количества и состава партий каждого типа данных, обрабатываемых в системе;
* разработать метод формирования эффективных расписаний обработки групп партий;
* исследовать разработанные методы и сделать выводы об эффективности разработанных алгоритмов и областях возможного применения системы в целом.

## Анализ существующих методов построения комплексных расписаний обработки партий данных

На современном этапе развития направления теории расписаний и задач оптимизации рассматриваются как точные, так и приближенные методы получения решения по составам партий.

Точные методы позволяют получить гарантировано лучшее решение, но при увеличении размерности входных параметров системы задача становиться невыполнимой (стремительно растут размеры матриц решений). Точные методы по построению расписаний для нескольких приборов представлены в [3].

Приближенные методы решения задачи оптимизации составов партий данных носят случайный характер и за частую основаны на алгоритмах, использующих случайные события.

Современные методы теории расписаний позволяют формировать статические расписания обработки единичных данных разных типов при заданном количестве приборов в многостадийных обрабатывающих системах (в частности в конвейерных системах) с использованием различных критериев определения эффективных решений. В работах [4, 5, 6] выполняется решение классических задач теории расписаний обработки единичных данных для одного либо нескольких сегментов конвейера при различных видах критериев оптимизации и наличии ограничений на директивные сроки окончания обслуживания. При этом развиваются как точные (ветвей и границ, ветвей и отсечений), так и приближенные методы получения расписаний выполнения программ обработки данных.

В работе [7] рассматриваются методы оптимизации расписаний обработки требований разных типов, совмещенных в группы (партии) для уменьшения времени переналадки и директивного срока окончания обслуживания. В основе всех представленных в [7] работ используется метод ветвей и границ для построения расписания обработки данных, объединенных в партии, и доказывается невозможность разрешения NP-трудности задачи при увеличении весовых коэффициентов (увеличения количества построенных партий разных типов данных). В работе проводится сравнительный анализ алгоритмов построения расписания обработки партий данных методом ветвей и границ и генетическим алгоритмом.

В работе [8] рассматривается задача планирования N заданий на одной машине, где требования обрабатываются в пакетном режиме, а время обработки каждого задания является простой линейно возрастающей функция в зависимости от времени ожидания задания, которое является временем между началом обработка партии, к которой принадлежит работа и начало обработки самой работы. В зависимости от выбора функции можно предположить, что для требований в пакете будет использована одна функция для упрощения задачи вычисления времени ожидания. В работе [8] так же используется эвристический метод ветвей и границ.

Проанализировав все вышеперечисленные источники можно сделать вывод, что большинство задач теории расписания решаются при помощи построения новых эвристик и использования эвристического подхода ветвей и границ. Этот метод позволяет найти точное решение задачи за фиксированное количество шагов. Но при увеличении числа обрабатываемых требований и приборов неуклонно растет размерность матриц расчётных переменных и количество шагов. Для многих NP-трудных задач алгоритм ветвей и границ не может быть применим, так как поиск решения будет занимать огромное количество времени и грандиозные объемы памяти, что невозможно при фиксированных параметрах современной вычислительной техники.

Отсюда можно сделать вывод, что текущая решаемая задача актуальна, так как не один из перечисленных методов не удовлетворяет условиям поставленной задачи. Отсюда следует, что необходимо сформулировать методы оптимизации партий данных и расписаний их обработки и на практике показать их применимость.

## Обоснование подхода к решению задачи построения расписаний обработки партий данных разных типов при наличии ограничения на интервал функционирования системы

Задача построения расписания групповой обработки партий при наличии ограничений имеет обобщённую цель – обработать максимальное количество данных. В то же время необходимо учитывать ограниченность временных интервалов обработки групп. Данная задача трудноразрешима. В соответствии с подходом вертикальной декомпозиции описанной в [9] достижение цели может быть представлено в виде иерархии подцелей. Тогда при разбиении обобщённой цели на подцели имеют место следующие свойства иерархической обработки:

* приоритетность решений, это свойство следует из необходимости передачи данных с уровня на уровень;
* зависимость эффективного решения на вышестоящем уровне от решения на нижестоящем уровне;
* достижение обобщённой (внешней, глобальной) цели системы возможно только при достижении всех подцелей (внутренних, локальных целей).

В результате декомпозиции обобщённой цели сформирована (определена) трёхуровневая иерархически упорядоченная структура поиска решения задачи (с локальными подцелями на каждом уровне), изображённая на рисунке 1.1.

Для оценки эффективности решений на каждом уровне должны быть введены критерии оценки, которые должны учитывать:

* на третьем (нижнем) уровне – эффективность использования оборудования конвейерной системы при размещении рассматриваемой партии (добавляемой в расписание для соответствующей группы) в последовательностях ;
* на втором (среднем) уровне – общую эффективность использования оборудования конвейерной системы при обработке всех партий данных группы  (анализ сформированного состава группы партий на основе построенного для неё расписания с точки зрения эффективности использования временного ресурса системы с учётом ограничений на длительность обработки);
* на первом (верхнем) уровне – общее количество данных, обработанных в системе в течение Z интервалов времени.

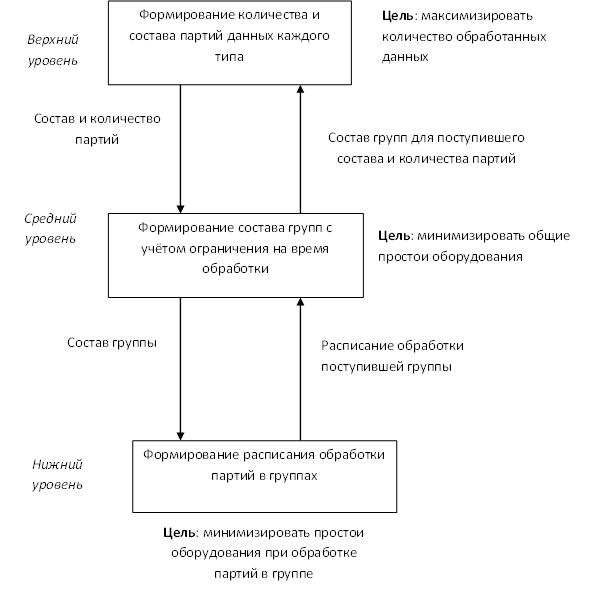


Рисунок 1.1 – Структурная схема системы построения расписаний   
обработки партий данных

## Обоснование использования аппарата теории иерархических и неантагонистических игр при построении расписаний групповой обработки партий данных

В п. 2.2 структура решения обосновывается как иерархически-упорядоченная система, на каждом уровне которой существует критерий оценки эффективности. Тогда в общем виде модель системы может быть формализована в виде:

,

где – решения на первом, втором и третьем уровнях соответственно;

– множества решений на уровнях,  – функции, с использованием которых задаются ограничения при выборе решений на этих уровнях;

– эффективные решения, полученные на втором и третьем уровнях соответственно.

При решении подобных задач используют понятия иерархических игр или игр по Штакельбергу [10]. Задачи с иерархическими играми относятся к разделу теории игр.

В игре по Штакельбергу участники выбирают стратегии последовательно; тот, кто анонсирует стратегию первым, называется лидером. Равновесие по Штакельбергу достигается, когда лидер решает задачу оптимизации своей целевой функции, принимая во внимание оптимальный ответ последователя.

В системе уровни (участники) формируют стратегии последовательно, вышестоящий уровень (лидер) решает задачу оптимизации целевой функции основываясь на оптимальном решении нижестоящего уровня (последователя). Это является классической формулировкой равновесия по Штакельбергу [10].

Для второго уровня системы применяется жадная стратегия наполнения групп партий данных и последующая оптимизация их составов. Взяв такой принцип за основу работы второго уровня можно установить, что решения, полученные на втором уровне, не взаимодействуют друг с другом (изменение критерия в одном решении не ведет к изменению решения в другом критерии).

Таким образом, решение задачи составления расписаний групповой обработки данных разных типов обеспечивается определением ситуации равновесия по Штакельбергу в иерархической игре (рисунок 1.2).

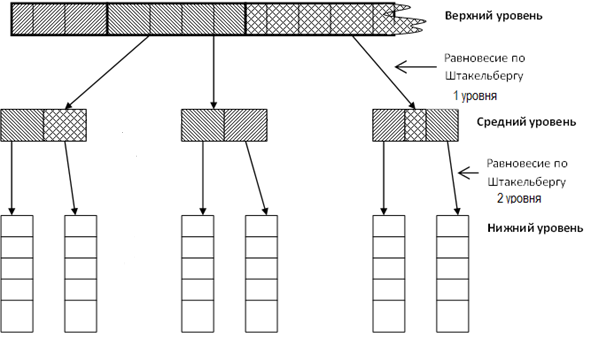


Рисунок 1.2 – Трёхуровневая организация системы построения расписаний   
обработки данных

## Выводы раздела 1

Таким образом, решение задачи составления расписаний групповой обработки данных разных типов при наличии ограничения на время функционирования системы можно представить в виде трехуровневой иерархической системы с заданием внутренних критериев на каждом из уровней функционирования системы.

# МОДЕЛЬ СИСТЕМЫ ПОСТРОЕНИЯ РЕШЕНИЯ ПО СОСТАВАМ ПАРТИЙ ДАННЫХ

## Описание иерархической модели построения комплексного расписания обработки партий данных

Для обоснования методов оптимизации составов партий, составов групп партий и расписаний обработки партий в рассмотрение введены обозначения и выполнены рассуждения, приведенные ниже [1]. Через *i* обозначен идентификатор типа обрабатываемых в системе данных, *n* – количество типов обрабатываемых в системе данных (),  – количество данных *i*-го типа, которые должны быть обработаны в системе. В постановке задачи являются заданным, что  (), тогда обрабатывающая программа должна быть выполнена в конвейерной системе  число раз, а для обработки однотипных данных формируются партии. Партия – это совокупность однотипных данных, обработка которых выполняется без переналадки сегментов конвейера. Партия является фиксированной, если в нее входят все данные *i*-го типа, либо партия может содержать не все  данных *i*-го типа, в этом случае в системе обрабатывается ни одна, а несколько партий данных этого типа. Через *l* обозначен индекс сегмента конвейерной системы, осуществляющего выполнение *l-*й части программы в системе (). Дисциплина обслуживания выполняемых в системе программ предполагает прохождение данными, которые они обрабатывают, всех сегментов конвейера, при этом если *l*-й прибор приступил к обработке данных *i*-го типа, обработка не может быть прервана. Все обрабатывающие приборы конвейерной системы характеризуются равными и неизменными во времени значениями производительности их работы. Выполнение на каждом *l*-м сегменте назначенной ему части *i*‑ой программы характеризуется длительностью обработки данных.

Обработка партий данных разных типов выполняется в течение интервалов времени задаваемой длительности, для которых введены обозначения: – интервал времени, в течение которого может быть реализована обработка партий данных. Т.к. обработка партий данных *i*-ых типов () реализуется в течение заданных интервалов времени функционирования системы  (), тогда на основе решения по количеству и составам партий данных различных типов формируются группы партий, каждая группа партий обрабатывается в течение одного из интервалов  (формируется *Z* групп партий). Группа партий – это совокупность партий, обрабатываемых в течение одного интервала времени  функционирования системы. Т.к. заданными являются ограничения на интервалы времени функционирования системы  (), тогда не все сформированные партий могут быть включены в состав групп партий. Состав групп партий для интервалов  () определяется таким образом, чтобы обеспечить максимальное количество обработанных партий данных каждого типа.

Выполнена декомпозиция обобщенной цели функционирования системы на совокупность иерархически упорядоченных целей подсистем (функций, решаемых на иерархически упорядоченных уровнях системы построения расписаний обработки партий данных). Формирование решений осуществляется на уровнях системы следующим образом [1]: первый – решения по составам партий данных, второй – решения по составам групп партий; третий – решения по порядку обработки на сегментах конвейера партий, входящих в группы.

Для формирования решений по составам партий данных *i*-ых типов () в рассмотрение введены обозначения: – количество партий данных *i*-го типа   
(), формируемых на первом уровне принятия решений, элементы  образуют вектор (*М*); (*А*)– матрица, элемент  которой – это количество данных *i*-го типа в *h*-ой партии (). Решение, формируемое на первом уровне системы имеет вид: [(*М*), (*А*)], где (*М*)– вектор количества партий данных *i*-ых типов, (*А*)–матрица количества данных в партиях.

Для формирования решений по составам групп партий введены обозначения:  () – группа партий, обрабатываемых в течение одного из интервалов (); – количество партий данных *i*-го типа в группе партий ;  – вектор количества данных *i*-го типа в  партиях в группе . Тогда партии данных *i*-го типа, входящие в группу партий , определены с использованием набора параметров вида: , а группа партий – совокупность таких наборов вида: , где – количество типов данных, партии которых входят в . Решение, формируемое на втором уровне системы – совокупность групп партий, имеет вид: {()}.

В соответствии с полученным решением по составу групп партий требуется определить порядок обработки партий каждой из групп на приборах системы (должна быть определена последовательность запуска партий каждой из групп на обработку на сегментах конвейера), т.е. расписание обработки партий соответствующей группы. Расписание обработки партий *z*-ой группы обозначено как , оно представляет собой совокупность последовательностей  запуска партий на обработку на *l*-ых сегментах конвейера (). Расписание  имеет вид: . Расписание обработки партий  для группы  формируется в предположении, что порядок обработки партий данных является одинаковым на всех *L* сегментах конвейера. Для формализации вида последовательностей  расписания  в рассмотрении введена матрица порядка обработки партий . Элемент , если партия данных *i*-го типа занимает в последовательности  *j*-ю позицию,  в случае, если партия данных *i*-го типа не занимает в последовательности  *j*-ю позицию, размер матрицы , где  – количество типов данных в партиях в группе ,  – число партий в последовательностях для группы . Порядок обработки партий группы на всех сегментах одинаков, поэтому достаточно определения одной матрицы порядка . В рассмотрение введена матрица – матрица количества данных *i*-го типа в партиях, занимающих в последовательности  *j*-е позиции (элемент  равен количеству данных *i*-го типа в партии, занимающей *j*-ю позицию в , размер матрицы   
). Решение, формируемое на нижнем (третьем) уровне иерархии подсистем в системе, имеет вид:

{}.

В силу наличия ограничений на время функционирования системы не все сформированные партии данных могут быть размещены в группах партий (могут быть обработаны в течение заданных интервалов ()), а цель функционирования системы состоит в обработке данных таким образом, чтобы было обработано максимальное количество требований, входящих в партии.

Для определения вида модели вычислительного процесса обработки партий данных, включенных в группы () введены следующие обозначения:  – время обработки данных *i*-го типа на *l*-ом сегменте конвейера (); – время переналадки *l*-го сегмента с обработки данных *i*-го типа на обработку данных *k*-го типа; – время первоначальной наладки *l*-го сегмента на обработку данных *i*-го типа;  – время начала обработки партии данных i-го типа, занимающей в последовательности  на *l*-ом сегменте *j*-ю позицию; – матрица моментов времени начала обработки партий данных *i*-ых типов, занимающих в  *j*-е позиции (для группы партий ); – матрица моментов времени начала обработки *q*-ых данных в партии, занимающей в  *j*-ю позицию (*q* – порядковый номер данных в партии в *j*-ой позиции в  (, где – количество данных в партии, входящей в группу , занимающей *j*-ю позицию в ). Элементы матрицы  определяются следующим образом: , где , – момент времени начала обработки первых данных в партии, занимающей *j*-ю позицию в , – количество типов данных, партии которых входят в одну рассматриваемую группу партий , – количество партий разных типов в группе . В рассмотрение введена матрица переналадок (), элементы  которой соответствуют длительностям переналадки сегментов с обработки данных *i*-го типа на обработку данных *k*-го типа, элементы – время первоначальной наладки сегментов на обработку данных *i*-го типа. Определение значений  и  (; ; ) выполняется в соответствии с изложенным ниже подходом для партий данных, входящих в одну группу  (индекс *z* группы партий  для временных параметров модели опущен при изложении).

Для позиции (*q*=1) данных *i*-го типа в (*j*=1)-ой партии в  выражение для определения  формируется при учете параметра  –времени наладки сегмента на обработку данных *i*-го типа: . Для  (,) имеем [1]: . В этом выражении первое слагаемое определяет длительность наладки сегмента конвейера на обработку данных в первой позиции в , второе слагаемое–длительность обработки данных в первой партии, предшествующих данным в *q*-ой позиции. Для первой позиции данных (*q*=1) (*j*=2)-ой партии в для  имеем: если  – время переналадки первого сегмента конвейера с обработки данных *i*-го типа (первая позиция партии в ) на обработку данных другого типа (вторая позиция партии в ), а – количество данных *i*-го типа в партии в первой (*j*=1) позиции в , тогда , где значение  определяется следующим образом:

, где .

Первой слагаемое в выражении для  и  определяет длительность наладки сегмента на обработку данных в партии, занимающей первую позицию в , второе слагаемое – длительность обработки данных в первой партии в , третье слагаемое – длительность переналадки сегмента с обработки данных в первой партии в на обработку данных во второй партии в . Для позиций    
() значения  определяются выражением вида:

, .

По аналогии обобщенное выражение для определения  при  для последовательности  имеет вид: , моменты времени начала обработки данных в *q*-ой позиции в партии с индексом : . Выражения для  и  сформированы в общем виде следующим образом [1]:

;

;

.

Использование выражений для ,  и ,  () позволяет определить временные характеристики вычислительного процесса выполнения конвейеризированных программ обработки данных на сегментах конвейера. Метод построения расписаний обработки партий предполагает добавление текущей рассматриваемой партии данных *i*-го типа в конец последовательностей    
() и определение эффективного местоположения партии в этих последовательностях. Тогда местоположение рассматриваемой партии в  может быть охарактеризовано текущими (для данного количества партий в  ()) простоями сегментов конвейера при обработке партий, нахо­дящихся в последовательностях .

В работе [1] выполнено формирование вида критерия эффективности принятия решений при построении расписаний обработки партий данных на нижнем уровне иерархии подсистем. Полученный критерий эффективности определения решения по порядку обработки партий в системе имеет вид [1]:

.

Первое слагаемое позволяет определить суммарный простой всех L сегментов конвейера перед началом обработки данных в первой позиции (q=1) в первой партии (*j*=1) в  (). Второе слагаемое определяет суммарный простой сегментов конвейера при переходе от обработки партии данных одного типа   
(в (*j*-1)-ой позиции в ) к обработке партии данных другого типа (в *j*-ой позиции в ). Третье слагаемое позволяет определить суммарный простой сегментов, вызванный ожиданием готовности данных при обработке их внутри партий.

Т.к. цель функционирования системы в течение заданных временных интервалов предполагает обработку данных таким образом, чтобы обеспечить максимальное количество обработанных требований, тогда полученное выражение использовано в качестве критерия эффективности решений  по составам групп партий на втором уровне иерархии системы [1].

На первом уровне иерархии критерий эффективности решений должен учитывать решение вида , формируемое на этом уровне, и эффективное решение по составам групп партий, формируемое на втором уровне в виде . Решение  характеризуется количеством обработанных партий. При определении значения критерия , характеризующего решение на первом уровне, используется само это решение. В качестве входных параметров при определении составов партий задается количество  () данных каждого типа, которые необходимо обработать за заданные промежутки времени . Тогда решение  может быть охарактеризовано количеством данных, которые должны быть обработаны. В соответствии с решением количество данных *i*-го типа, которые должны быть обработаны в системе, определяется выражением, где – элемент матрицы (*A*), значение которого равно количеству данных *i*-го типа в *h*-ой партии (), – количество партий данных *i*-го типа – *i*-ый элемент вектора (*M*). Общее количество данных различных типов, которые должны быть обработаны в системе, в соответствии с решением  определяется выражением .

Значение  – это количество данных в партии, занимающей *j*-ю позицию в последовательностях  (). Эти рассуждения распространены на последнюю партию в последовательности . Для определения последней партии, а также данных в последней позиции в этой партии использованы обозначения: –количество партий, входящих в группу  (индекс последней партии   
в ); – количество данных, входящих в последнюю в  партию (индекс последних данных в этой партии). Тогда – момент времени начала обработки последних данных в партии с индексом  на *L*-ом сегменте конвейера, а время окончания обработки этой партии на этом сегменте определяется выражением вида: . Ограничение на время обработки партий данных, входящих в группы  () , имеет вид:при .

Тогда модель многоуровневого программирования для определения эффективных составов партий, составов групп партий и расписаний обработки партий имеет вид [1]:

1) первый (верхней) уровень: , где

;

2) второй уровень: , где

;

3) третий уровень:  (), где

;

4) ограничения на длительность реализации обработки партий групп    
():

 при .

С учётом модели процедура определений эффективных составов партий данных, составов групп партий, обрабатываемых в течение задаваемых интервалов времени функционирования системы (), расписаний обработки партий каждой из групп в конвейерной системе предполагает:

1) определение количества партий данных *i*-ых типов () и составов этих партий (распределение данных *i*-ых типов по партиям);

2) определение составов групп партий данных, обрабатываемых в течение задаваемых интервалов времени ();

3) формирование порядка обработки партий (расписания их обработки).

Иерархичность функций в системе определяет порядок принятия решений:

1) решение по составам партий данных в виде [(*M*), (*A*)], где (*М*) – вектор количества партий данных различных типов, (*А*) – матрица количества данных в партиях;

2) решение по составам групп партий в виде {()};

3) решение по порядкам обработки партий данных, входящих группы   
() в виде:  ().

## Обоснование способа формирования решения по составам партий данных заданного типа

Для обоснования способа формирования решений по составам партий данных в количестве  в рассмотрение введены следующие условия и рассуждения:

1) количество данных -го типа () в формируемых партиях не может быть меньше, чем 2, т.е. (), где *h* – идентификатор партии данных (номер столбца в -ой строке матрицы А); если при формировании начального решения по составам партий данных -го типа для задаваемого их количества  для *h*=1 (первая партия) получено, что , тогда исследование эффективности составов партий данных -го типа в количестве  должно быть завершено, т.к. количество данных в партиях не может быть менее 2;

2) способ формирования начального решения для заданного количества  партий данных -го типа предполагает, что (, ), а элемент  определяется следующим образом:; в дальнейшем при построении решений по составам партий данные извлекаются из партии с *h*=1 и распределяются по остальным партиям ();

3) задаваемые первоначально количества  партий данных *i*-ых типов   
() равны 2 (); модификация количества партий данных рассматриваемого -го типа предполагает, что параметр  увеличивается (при определении составов партий) до тех пор, пока в начальном решении по составам партий для полученного значения  выполняется условие ; при выполнении условия  (для начального решения при соответствующем значении ) исследование составов партий -го типа прекращается;

4) формирование эффективных решений по составам партий (в количестве ) связано с увеличением (на 1) количества данных в партии с индексом  (индекс столбца в соответствующей строке матрицы *А*) при одновременном уменьшении количества данных в партии с *h*=1 (последовательное увеличение количества данных в партиях с индексом ); решение по составам партий в количестве  характеризуется значением целевой функции , для которого возможно выполнение условия  (отрицательное значение левого дискретного градиента целевой функции ) – количество данных в партии с индексом  увеличивается, что приводит к уменьшению значения ;

5) формирование эффективных решений по составам партий предполагает увеличение числа данных в партии с индексом  (рассматриваемый номер партии, при неизменном составе остальных партий с индексами *h* )) и уменьшении количества данных в партии с *h*=1 в соответствии с выражением ; при выполнении условия   модификация составов партий может быть продолжена; при выполнении условия  полученное решение по составам партий в количестве  является решением, аналогичным полученному на предыдущих шагах алгоритма (дублирует полученное ранее решение), тогда полученное решение, для которого выполняется введенное условие, не интерпретируется.

Сформулированный способ модификации составов партий -го типа предусматривает, что на основе решений, находящихся в окрестности текущего локально оптимального решения с меньшей метрикой (окрестность ), формируются решения, находящиеся в окрестности с большей метрикой (окрестность ). Метрика *k* окрестности  определяется выражением вида , где – элемент -ой строки матрицы , соответствующей текущему локально оптимальному решению по составам партий данных -го типа, сформированному на s-ой итерации алгоритма, – элемент -ой строки матрицы , соответствующей решению по составам партий, сформированному на (s+g)-ой итерации алгоритма, которое находится в окрестности  с метрикой *k* решения .

В матрице  реализуется хранение промежуточных решений, находящихся в окрестности , на основе которых реализуется формирование решений, входящих в окрестность  (в матрице  хранятся решения, используемые при формировании новых решений в окрестности ). В матрице  реализуется хранение формируемых решений, находящихся в окрестности . Тогда на основе промежуточного решения по составам партий данных (-ой строки  матрицы ), формируется совокупность новых решений, хранимых в матрице , находящихся в окрестности . Если для текущего рассматриваемого решения (-ой строки  матрицы ) выполняется условие , тогда увеличение на 1 значения  (при неизменном значении ), а затем увеличение на 1 значения  (при неизменном значении ) позволяет получить одинаковые решения по составам партий данных -го типа. Сформулированное условие позволяет исключить повторяющиеся решения, получаемые при модификации решения по составам партий данных -го типа. На его основе сформулирована Теорема 2, ограничивающая количество формируемых решений.

В соответствии с введенными в рассмотрение типами окрестностей, предполагается, что решения, входящие в окрестность первого типа, формируются путем изменения составов партий в количестве  (при неизменном значении ). Решения, входящие в окрестность второго типа, формируются путем увеличения количества партий , задания для количества партий  начального решения и последующего изменения составов партий для этого количества.

## Метод построения решения по составам партий данных заданного типа данных

Формулируемый метод позволяет выполнить локальную оптимизацию решения (в рамках его окрестностей разных типов). Для обоснования формулировки алгоритма определения эффективных составов партий в рассмотрение введены: 1) способ формирования решений по составам партий данных некоторого -го типа; 2) способ формирования для текущего локально оптимального решения окрестностей первого типа, в рамках которых будет выполняться поиск эффективных (лучших) решений.

Рассмотренный метод основан на жадном подходе. Из этого следует, что полученные решения будут максимальны в формируемой окрестности на текущем шаге алгоритма.

## Описание метода формирования решения по составам партий заданного типа данных

Для определения локально оптимального решения по составам партий данных *i*-ых типов () сформулированы: 1) способ формирования решений по составам партий данных некоторого -го типа, включаемых в окрестности  первого типа (предполагающий изменение составов партий данных при их количестве ), на основе решений, входящих в окрестность ; 2) обобщенный алгоритм идентификации локально оптимального решения по составам партий данных. Входными параметрами алгоритма формирования решений, включаемых в окрестность  первого типа, на основе решений, входящих в окрестность , являются: идентификатор -го типа данных, для которых определяется составы партий; матрица  размерностью , содержащая решения, входящие в окрестность текущего локально оптимального решения; количество партий , составы которых будут изменяться; количество  решений в окрестности ; индекс s текущего шага алгоритма идентификации локально оптимального решения.

Алгоритм формирования решений по составам партий данных -го типа в окрестности  на основе решений в окрестности  следующий:

1) инициализация значений параметров , , , , ;

2) инициализация номера партии , состав которой будет изменяться, значением 2 ();

3) модификация значения ; для -ой партии -го типа увеличение количества данных на 1; формирование решения по составам партий (строки  матрицы ): (),, ; ;

4) проверка выполнения условия  (при заданном количестве партий ); в случае выполнения этого условия сформированное в виде строки  матрицы  решение не интерпретируется (в соответствии с Теоремой 1), тогда ; выполняется задание значения параметра *j*=1 (*j* – значение шага изменения номера партии ) и реализуется переход на шаг 5;

5) осуществляется проверка условия ; при  реализуется проверка условия формирования решения по составам партий -го типа в виде  (в соответствии с Теоремой 2); если условие  не выполняется, тогда реализуется переход на шаг 6; при выполнении условия существует возможность формирования дополнительного решения по составам партий данных -го типа, тогда , осуществляется переход на шаг 3;

6) в случае выполнения условия  (в соответствии с Теоремой 2) при модификации ()-ой партии будет получено решение, дублирующее полученные ранее; осуществляется модификация параметра *j*=*j*+1; выполняется переход на шаг 5;

7) при выполнении  сформированы все возможные решения (хранящиеся в матрице ) с использованием одного решения с индексом    
(-ой строки  матрицы ); переход на шаг 8;

8) модификация текущего значения – индекса решения, хранимого в матрице  (осуществляется переход к следующему решению в окрестности , на основании которого могут быть сформированы составы партий данных): , если  (рассмотрены не все решения из окрестности , хранимые в матрице ), тогда реализуется переход на шаг 2; в случае выполнения условия  все решения, хранимые в матрице , находящиеся в окрестности  текущего локально оптимального решения, использованы для формирования решений, находящихся в окрестности , хранимых в матрице ; выполняется переход на шаг 9;

9) при  выполняется сравнение решений, хранимых в матрице , с точки зрения дублирования ими друг друга (процедура сравнения предполагает формирование копии матрицы , в копии матрицы  упорядочивание элементов в каждой -ой строке () по убыванию значений элементов, поэлементное сравнение каждой -ой строки () с другими   
-ыми строками (), удаление из матрицы  строк с индексами , которые соответствуют строкам в копии матрицы , дублирующим рассматриваемую -ю строку в копии ; при удалении q2’-ой строки из матрицы  реализуется изменение ); в результате повторяющиеся решения из матрицы  удалены, остальные решения, хранящиеся в этой матрице (за счет ее дублирования в процедуре сравнения), остались без изменения; инициализация значений  ; ; при  осуществляется переход на шаг 11;

10) реализация проверки условия ; при  не сформированы новые решения в окрестности , которые могут быть более эффективными, чем рассматриваемое локально оптимальное решение *A*(*s*), т.е. не сформирована окрестность  локально оптимального решения, выполняется переход на шаг 16;

11) рассмотрение решений по составам партий данных -го типа, находящихся в окрестности  текущего локально оптимального решения, с точки зрения их эффективности; для этого выполняется инициализация -ой строки матрицы А’ решением по составам партий данных, хранимым в -ой строке  матрицы : (); передача сформированного решения  на второй уровень для определения составов групп партий (при этом исходное локально оптимальное решение *A*(*s*) не изменяется);

12) получение со второго уровня иерархии эффективного решения по составам обрабатываемых в группах, в виде матрицы , использование решения  для вычисления значения критерия  (значения );

13) для сформированного решения  вычисление левого (либо правого) дискретного градиентов  (); вычисление    
() выполняется по формулам [2]: а) ; б) ;

14) при выполнении условия  сформированное решение  не является более эффективным (при минимизации критерия   
), чем решение (для количества партий ); осуществляется модификация значения ; если  (т.е. рассмотрены не все решения, находящиеся в окрестности ), тогда осуществляется модификация индекса шага алгоритма *g*=*g*+1, реализуется переход на шаг 11 (переход к анализу эффективности следующего решения из окрестности ); если , тогда выполняется переход на шаг 16;

15) если для текущего рассматриваемого решения по составам партий   
(-ой строки  матрицы ) реализуется , тогда выполняется сравнение значения  с максимальным значением  градиента, достигаемым в рассматриваемой окрестности  (для данных -го типа); если , тогда текущее решение по составу партий -го типа данных является наилучшим среди рассмотренных решений из окрестности , индекс решения (строки  матрицы )  сохраняется: , модификация значения : ; реализуется переход к следующему решению в окрестности , для этого ; если  (рассмотрены не все сформированные решения, находящиеся в окрестности ), тогда *g*=*g*+1 и реализуется переход на шаг 11 (переход к анализу эффективности следующего решения из окрестности ); если  (рассмотрены все решения по составам партий данных -го типа, находящиеся в окрестности ), тогда выполняется переход на шаг 16;

16) останов алгоритма.

Выходными параметрами алгоритма определения эффективных решений в окрестности  являются: значение  количества сформированных решений по составам партий данных -го типа, входящих в окрестность  (при  на основе решений, входящих в окрестность , могут быть сформированы новые решения, входящие в окрестность  первого типа (при неизменном количестве партий )), при  на основе решений, входящих в окрестность , не определены решения по составам партий данных -го типа в , на основе которых могли быть сформированы решения, входящие в окрестность  первого типа, поэтому требуется выполнить переход к формированию окрестности второго типа путем увеличения значения  и определения для этого количества партий  начального решения по их составу; значение – идентификатора строки в матрице , соответствующей наилучшему решению по составам партий данных -го типа в сформированной окрестности  (при  и  не получено решение, входящее в окрестность , являющееся лучшим решением по составам партий данных -го типа, чем текущее локально оптимальное решение, необходим переход к окрестности  первого типа с большей метрикой для определения в ней эффективного решения по составам партий данных -го типа); матрица , содержащая (при ) все решения, входящие в окрестность   
. Реализация предложенного алгоритма формирования решений по составам партий данных -го типа, входящих в окрестности  с различным значением метрики *k*, представлена на рисунке 2.2 для заданных значений параметров  и .

На рисунке представлено дерево построения решений согласно алгоритму формирования решений по составам партий данных заданного типа.

Расшифровка рисунка представляет:

1) не помеченные - интерпретируемые решения;

2) \_\_\_\_ – решения, исключаемые из рассмотрения в соответствии с Теоремой 1;

3) .\_.\_.\_. – решения, которые не формируются в соответствии с Теоремой 2;

4)…….– решения, исключаемые из рассмотрения как дублирующие решения в окрестности .



Рисунок 2.1 – Реализация алгоритма формирования решений по составам партий в окрестностях  (при  и )

## Метод построения решения по составам партий различных типов данных

Формулируемый метод позволяет выполнить локальную оптимизацию решения (в рамках его окрестностей разных типов). Для обоснования формулировки алгоритма определения эффективных составов партий в рассмотрение введены: 1) способ формирования решений по составам партий данных некоторого -го типа; 2) способ формирования для текущего локально оптимального решения окрестностей первого типа, в рамках которых будет выполняться поиск эффективных (лучших) решений.

## Обоснование метода построения решения по составам партий различных типов данных

Для обоснования метода формирования эффективных составов партий данных в рассмотрение введены следующие обозначения: – идентификатор рассматриваемого типа данных, состав партий которого изменяется (модифицируется) на текущем шаге алгоритма; – количество партий данных рассматриваемого -го типа, состав партий которого определяется; s– индекс исходного решения, подлежащего оптимизации путем поиска нового более эффективного решения в рамках некоторой окрестности; *g* – индекс шага алгоритма, выполняемого по отношению к номеру шага *s*, который соответствует новому формируемому решению, находящемуся в окрестности  ((*s+g*)– номер шага алгоритма по формированию решения в окрестности  текущего локально оптимального решения); – индекс (номер) партии, состав которой изменяется (модифицируется) при реализации алгоритма на текущей его итерации (на промежуточной итерации алгоритма выполняется изменение состава партии данных с индексом (номером) , итерация алгоритма предполагает реализацию определенной последовательности его шагов); *I* – множество типов данных, для которых должно быть выполнено формирование составов партий (первоначальный состав этого множества имеет следующий вид: ); – копия множества *I* типов данных, используемая при формировании локально оптимального решения по составам этих партий; *A’*– матрица – аналог матрицы составов партий *А*, используемая при определении наилучшего решения в окрестности  текущего локально оптимального решения, которому соответствует матрица A (матрица *А’* используется при формировании новых промежуточных решений);  и – матрицы, предназначенная для хранения (буферизации) составов партий данных -го типа, сформированных на различных итерациях алгоритма; – количество решений по составам партий данных -го типа, сформированных на (s+g)-ой итерации алгоритма (хранение решений реализуется в матрице  размерностью ); – количество решений по составам партий данных -го типа, полученных на последующей ((*s+g*)+1)-ой итерации алгоритма (хранение решений реализуется в матрице  размерностью ); – индекс решения по составам партий данных -го типа в матрице  (); – индекс решения по составам партий данных -го типа в матрице  (); – параметр, предназначенный для хранения идентификатора решения (номера строки матрицы ), гарантирующего максимальное по модулю значение левого дискретного градиента целевой функции ; *G*– максимальное по модулю значение дискретного градиента  [2], достигаемое в окрестностях текущего локально оптимального решения; – параметр алгоритма, реализующий буферизацию значения , используемый при определении наилучших решений по составам партий некоторого -го типа; – множество типов данных, для которых .

## Описание метода формирования составов партий различных типов данных

Метод формирования составов партий данных реализует поиск локально оптимального решения в рамках окрестностей, способ формирования которых формулируется ниже. Для текущего локального оптимального решения по составам партий данных *i*-ых типов рассматриваются два типа окрестностей. При формировании окрестности первого типа построение решений связано с изменением составов партий данных в рассматриваемом их количестве  (), при формировании окрестности второго типа построение решений связано с увеличением количества  партий данных *i*-го типа и заданием начального решений для этого количества партий. Таким образом, если построение решений путем изменения составов партий данных -го типа в количестве  является невозможным (выполнены условия окончания формирования составов партий данных -го типа в количестве ), тогда реализуется изменение количества  партий данных этого типа, состав которых будет определяться.

## Метод формирования составов партий различных типов данных

На основе рассмотренного способа формирования решений по составам партий данных некоторого -го типа, входящих в окрестность  первого типа для текущего локально оптимального решения , сформулирован алгоритм определения локально оптимального решения по составам партий данных *i*-ых типов () в соответствии с введенным критерием . Алгоритм определения локально оптимального решения по составам партий данных *i*-ых типов () содержит следующий порядок шагов:

1) задание для каждого *i*-го типа данных значений параметров    
() равными 1 (– количество решений по составам партий данных *i*-ых типов, сформированных на текущей итерации алгоритма, хранение которых осуществляется в матрице (соответственно, храниться в матрице ), задается равным 1);

2) формирование начального решения (составов партий данных *i*-ых типов ()) следующим образом: а)  (); б) инициализация *h*-ых элементов *i*-ых строк матрицы (*А*) на основе выражений вида:   
(), ; в) проверка условия корректности начального решения по составам партий данных -ых типов ():  и  ; г) если при построении начального решения для какого-либо *i*-го типа данных () условие его корректности по составам партий не выполняются, тогда для этого типа данных формируется одна фиксированная партия: , , ; при этом формируется новый состав множества *I* типов данных следующим образом: ; параметр *G* инициализируется значением 0; инициализация множества :; определение значения  для сформированного начального решения  по составам партий данных (текущего локально оптимального решения);

3) задание -го типа данных, составы партий которого определяются: ;

4) проверка реализации условия , в случае выполнения условия осуществляется инициализация значений параметров , *k*=0 и элементов матрицы  для -го типа данных  (,) (буферизировано решение по составам партий данных -го типа, подлежащее оптимизации);

5) определение решений по составам партий данных -го типа, формируемых на основе решений, находящихся в матрице , и входящих в окрестность  первого типа текущего локально оптимального решения  (для этого вызывается алгоритм формирования составов партий данных -го типа, рассмотренный выше, использующий  решений, хранящихся в матрице , для формирования новых решений в окрестности  и определяющий наилучшее решение среди полученных в окрестности );

6) реализация проверки условия  для параметра, полученного из алгоритма формирования решений в окрестности , при  для -го типа данных не сформированы новые решения в окрестности , которые могут быть более эффективными, чем рассматриваемое локально оптимальное решение, т.е. не сформирована окрестность  локально оптимального решения, поэтому никакие решения не могут быть проанализированы и на их основе не могут быть сформированы новые решения; выполняется переход на шаг 7; при выполнении условия  осуществляется переход на шаг 14;

7) при  реализуется формирование окрестности второго типа текущего локально оптимального решения  (решение в этой окрестности определяется путем увеличения количества  партий данных -го типа и формирования начального решения по составам партий для значения ), выполняется модификация количества партий: ; для количества партий данных  выполняется формирование начального состава (формируется решение, входящее в окрестность второго типа рассматриваемого локально оптимального решения): осуществляется инициализация значений ; , , , , элементов матрицы  для -го типа данных  (), ;

8) выполняется проверка корректности сформированного начального решения по составам  партий данных: ;  (); при выполнении условия корректности начальных составов  партий реализуется переход на шаг 9; если условие корректности начального решения по составам  партий данных не выполняется, тогда формирование новых решений по составам партий данных -го типа является невозможным, текущая -ая строка матрицы *A* соответствует локально оптимальному решению по составам партий данных этого типа; реализуется удаление -го типа данных из множества *I*: ; формирование решений по составам партий данных для -го типа данных прекращается, реализуется переход на шаг 14;

9) анализ эффективности начального решения по составам партий данных в количестве  (единственное решение в окрестности второго типа), для этого выполняется инициализация -ой строки матрицы *А’* решением по составам партий данных, хранимым в строке  матрицы : (); передача сформированного решения  на второй уровень для определения составов групп партий;

10) получение со второго уровня иерархии эффективного решения по составам групп, в виде матрицы , использование решения  для вычисления значения критерия  (значения );

11) для сформированного решения  вычисление левого (либо правого) дискретного градиентов  (); вычисление    
() выполняется по формулам [2]: а) ; б) ;

12) при выполнении условия  сформированное решение  не является более эффективным (при минимизации критерия ), чем решение  (для количества партий ), тогда для этого решения выполняется инициализация параметра  (равенство 0 свидетельствует об отсутствии решения в окрестности второго типа, лучшего, чем текущее локально оптимальное решение); выполняется переход на шаг 14 (при этом единственное сформированное решение в окрестности второго типа сохранено в матрице , параметр );

13) если для сформированного начального решения по составам  партий (строки  матрицы ) выполняется условие , тогда реализуется присваивание  (т.к. это единственное решение в окрестности  второго типа и нет необходимости его сравнения с другими решениями в этой окрестности), а также инициализация ;

14) осуществляется проверка условия ; в случае его выполнения реализуется переход на шаг 3; при  осуществляется переход на шаг 15;

15) проверка условия , при его выполнении отсутствуют типы данных (), для которых сформированные решения по составам партий удовлетворяют условию корректного их состава (т.е. отсутствует возможность определения и анализа новых решений), выполняется переход на шаг 20; при  реализуется инициализация множества  *i*-ых типов данных, для которых выполняется условие  и множества (); при выполнении условия  на предшествующих итерациях алгоритма сформированы лучшие решения по составам партий данных *i*-ых типов (), чем текущее локально оптимальное решение; при  реализуется переход на шаг 16; при  в окрестностях  первого типа либо в окрестностях второго типа локально оптимального решения  нет лучшего решения; выполняется переход на шаг 18;

16) задание -го типа данных, эффективность лучшего решения по составам партий которого сравнивается с эффективностью лучших решений для других *i*-ых типов (,): ;

17) осуществляется проверка условия ; при его выполнении в окрестности первого типа  либо в окрестности второго типа рассматриваемого локально оптимального решения  не найдено лучшего решения, реализуется переход на шаг 16 (переход к анализу решений для следующего типа данных); в случае выполнения  в окрестности первого либо второго типов получено более эффективное решение по составам партий данных -го типа, тогда реализуется сравнение значения  и значений  для лучших решений по составам партий данных *i*-ых типов (для всех и ); если  (для всех  и ), тогда значение  для лучшего решения по составам партий -го типа в окрестности  либо окрестности второго типа для  является максимальным по модулю среди значений градиентов  лучших решений по составам партий данных *i*-ых типов (и ), параметр  содержит индекс строки матрицы , в которой реализуется хранение решения по составам партий данных -го типа, являющегося наиболее эффективным среди решений по составам партий других i-ых типов (); при его выполнении реализуется модификация -ой строки в матрице А составов партий при неизменном виде *i*-ых строк этой матрицы ( и ):  при ; если условие  (для всех  и ) выполняется в случае, когда для текущего локально оптимального решения  рассматривается окрестность второго типа решений по составам партий -го типа, тогда реализуется модификация -го элемента вектора *M*; в результате зафиксировано новое локально оптимальное решение  по составам партий данных *i*-ых типов (); проводится инициализация множества : и переход на шаг 3;

18) идентификация -го типа данных, решения по составам партий данных которого буферизируются в матрице : , ; для -го типа данных реализуется присваивание:  (, , ), присваивание  (т.е. выполнена буферизация решений по составам партий -го типа, находящихся в рассматриваемой окрестности, для последующего формирования на их основе решений, находящихся в окрестности большего размера); в результате сформирована матрица , содержащая решения, на основе которых будут формироваться решения в окрестности  с большей метрикой;

19) проверка выполнения условия , если условие выполняется, тогда реализуется переход на шаг 18; в случае выполнения условия  осуществляется присваивание , *k*=*k*+1 и переход на шаг 3;

20) корректное окончание алгоритма.

## Анализ эффективности метода построения решения по составам партий различных типов данных

Анализ эффективности рассмотренного метода определения составов партий данных *i*-ых типов () при заданных длительностях интервалов времени функционирования системы возможен совместно с реализацией предложенных методов определения локально оптимальных составов групп партий, и расписаний обработки партий каждой из групп. В рамках предлагаемой работы наряду с методом определения составов партий формулируется метод определения эффективных (локально оптимальных) расписаний обработки партий, входящих в каждую из групп ().

## Выводы раздела 2

В данном разделе была рассмотрена модель трехуровневой системы. На основе модели были предложены и разработаны метод формирования составов партий всех типов данных и метод оптимизации составов партий фиксированного типа. На основе этих методов в разделе 4 описаны соответствующие программные модули и представлены их структурные схемы.

# МЕТОД ОПРЕДЕЛЕНИЯ ПОРЯДКА ОБРАБОТКИ ПАРТИЙ ДАННЫХ (РАСПИСАНИЙ ОБРАБОТКИ ПАРТИЙ ДАННЫХ)

## Обоснование метода определения порядка обработки партий данных заданных типов

В силу используемого предположения о том, что партии в сформированном расписании имеют одинаковый порядок обработки во всех последовательностях  () для отдельной рассматриваемой группы  и того, что количество типов данных ограничено, тогда является ограниченным и количество возможных решений. Для формирования расписаний обработки партий данных в системе использован метод окрестности (метод поиска лучшего решения в рамках некоторых окрестностей текущего локального оптимального решения). Для обоснования алгоритма определения эффективного порядка обработки партий данных в последовательностях  (), образующих расписание  отдельной группы , в рассмотрение введены следующие обозначения: s-индекс (номер) шага алгоритма, на котором зафиксировано текущее локально оптимальное решение по порядку обработки партий в группе ();  – количество партий данных, размещенных в последовательностях  () на предшествующих и текущем s-ом шагах алгоритма;  – вектор, *i*-ый элемент которого – это количество партий данных *i*-го типа, размещенных в последовательностях  () на предыдущих и текущем s-ом шагах алгоритма; g– количество выполненных шагов по изменению локально оптимального решения  для получения решения , соответствующего новому порядку партий в    
() (т.е. индекс шага алгоритма по определению лучшего решения в окрестности текущего локально оптимального); – метрика окрестности локально оптимального решения , используемая для идентификации отличия сформированного решения  от решения ;   
 – максимально возможная метрика окрестности исходного решения , в которой выполняется поиск более эффективного решения;  – окрестность решения , в которой выполняется поиск более эффективного решения;  – индекс элемента вектора , соответствующего составам партий данных *i*-го типа в группе , который содержит количество данных в партии, добавляемой в последовательности (), *v* – индекс текущего рассматриваемого столбца матриц  и , в *i*-ой строке которого значения элементов  и  соответствуют характеристикам рассматриваемой партии данных *i*-го типа, эффективное местоположение которой в последовательностях  определяется на текущей итерации алгоритма (, );  – индекс столбца матриц  и , в котором первоначально размещаются характеристики рассматриваемой партии данных *i*-го типа при добавлении этой партии в последовательности  (, ).

Вектор  используется при определении количества  партий данных различных типов, размещенных в последовательностях (). Первоначальная инициализация элементов () выполнена следующим образом:  (). Значение количества партий , размещенных в  (), определяется выражением: . Расчет метрики  окрестности решения : , где – количество типов данных, партии которых включены в рассматриваемую группу , – общее количество партий данных разных типов, размещенных в последовательностях  (). Метрика  определяет число элементов в матрице  (число позиций в последовательностях ), в которых изменены значения по отношения к виду матрицы , соответствующей локально оптимальному решению . Начальное решение  по определению позиции партии данных *i*-го типа в последовательностях () формируется путём добавления -ой партии данных *i*-го типа в количестве ) в конец этих последовательностей. Через *g* обозначен номер промежуточного решения , которое формируется на основе исходного , находящегося в окрестности , характеризуемой метрикой . Формирование решения предполагает изменение положения рассматриваемой партии в последовательностях  относительно ее положения, которое соответствует решению .

## Описание алгоритма формирования эффективного расписания групповой обработки партий данных

Алгоритм построения эффективного расписания обработки партий, входящих в группу , предполагает последовательное размещение в последовательностях () всех партии данных *i*-ых типов () из этой группы и идентификацию эффективного местоположения каждой из этих партий в . Таким образом, алгоритм реализует жадный подход к оптимизации, при котором каждое последующее эффективное решение по размещению одной из партий группы  в последовательностях  формируется на основе эффективных решений по порядку обработки партий, рассмотренных на предыдущих шагах алгоритма. Порядок шагов алгоритма построения эффективного расписания обработки партий рассмотрен с точки зрения размещения в последовательностях () партий данных некоторого *i*-го типа в количестве  и определения эффективного положения (эффективных позиций) этих партий в последовательностях  (). Алгоритм определения эффективного положения (позиций) партий данных *i*-го типа в последовательностях  () имеет следующий порядок шагов:

1) некоторое сформированное на предыдущем (*s*-1)-ом шаге алгоритма решение определяется соответствующим видом матриц  и , номер партии  (номер элемента вектора  ()), данные которой в количестве  размещается в последовательностях ()), инициализируется значением 1 ();

2) в силу того, что – количество партий данных, добавленных в () до s-го шага алгоритма, тогда для партии данных *i*-го типа определяется индекс  столбца матриц ,, в котором будут размещаться параметры, ей соответствующие: ;

3) добавляемая в последовательности  партия данных *i*-го типа в количестве  элементов размещается в конце последовательностей (); для этого в *i*-ой строке -го столбца матриц  и  задаются значения, соответствующие этой партии (элементы , и  инициализируются значениями 1 и  соответственно); модификация матриц  и , связанная с инициализацией элементов -ых столбцов, выполняется следующим образом: ; ; ,  при  и ; сформированное начальное решение по размещению в последовательностях () рассматриваемой партии данных *i*-го типа фиксируется как локально оптимальное решение: ;

4) значение *g* индекса шага текущего промежуточного решения инициализируется значением 1 ((*s+g*) – номер промежуточного шага алгоритма, связанного с определением локального оптимального решения по размещению партии данных *i*-го типа в количестве  элементов в окрестности текущего решения ), индексу текущего рассматриваемого столбца v матриц  и  присваивается значение ;

5) на основе матриц ,  реализуется вычисление элементов матриц  ;

6) для полученного вида матриц ,  и матриц   определяется значение критерия  и значения элементов матрицы ;

7) в последовательностях  изменяется порядок партий таким образом, что рассматриваемая партия данных *i*-го типа перемещается на одну позицию в начало ; выполняемые действия с матрицами  и  имеют вид: , , , , , , , , где k – индекс строки в матрицах  и , в которой элементы  и  (предыдущая позиция для рассматриваемой партии, которую она займет на (s+g)-ом шаге алгоритма); в результате формируются модифицированные матрицы , ; индекс v столбца, идентифицирующий местоположение парии в  , модифицируется: ;

8) с использованием матриц ,  вычисляются матрицы и , а также значение критерия;

9) приполучено решение не худшее, чем текущее, тогда решение  фиксируется как локально эффективное: , фиксируется критерия ; значение *s*=*s+g*, значение индекса g шага поиска следующего локального оптимального решения в окрестности текущего  задаётся равным 1, выполняется переход к шагу 7;

10) если , тогда проверяется выполнение условия  (– метрика максимально возможной окрестности исходного решения , являющегося локально эффективным, в которой выполняется поиск более эффективного решения ;

11) при выполнения условия  реализуется дальнейший поиск более эффективного решения в окрестности  исходного расписания , модифицируются: индекс g промежуточного шага поиска решения (g=g+1), значение метрики окрестности ; проверяется выполнение условия  (рассматриваемая партия является первой в  () (, )); при истинности условия  выполняется переход к шагу 7, при выполнении условия  выполняется переход к шагу 13;

12) при выполнении условия  реализуется переход к шагу 13;

13) полученное локально эффективное решение фиксируется (фиксируется полученный в результате порядок партий), изменение положения рассматриваемой партии данных *i*-го типа в количестве  в дальнейшем не реализуется;

14) модификация значений количества партий данных *i*-го типа, которые должны быть размещены в последовательностях  : ; изменение значения счетчика количества размещенных в   партий данных *i*-го типа: ; если , тогда реализуется изменение индекса следующей рассматриваемой партии данных *i*-го типа: ; изменение значения шага алгоритма: s=s+1; изменение номера столбца матриц , , в котором на следующем шаге алгоритма будут размещаться значения параметров, характеризующих добавляемую в   партию данных:; выполнить переход к шагу 3;

15) при выполнении условия  в последовательностях   размещены все партии данных *i*-го типа, включенные в состав группы партий  в соответствии с решением .

Алгоритм формирования порядка обработки партий позволяет выполнить упорядочивание партий данных каждого *i*-го типа () в количестве  (в каждой из которых  данных ) в последовательностях  , т.е. определить расписание обработки партий на сегментах конвейера. Реализация алгоритма обработки партий данных должна быть выполнена совместно с реализацией алгоритма формирования составов партий данных и алгоритма определения групп партий, соответствующих их сформированным составам партий.

## Выводы раздела 3

В данном разделе был сформирован и описан метод установки порядка обработки партий данных в группах и расчета времени обработки группы партий данных. Этот метод, так же как и методы, описанные в разделе 2, формирует начальное решение и затем модифицирует его, для получения наилучшего решения на текущем шаге алгоритма. Так же этот метод должен проверить полученное решение на временное ограничение (хватает ли времени обработки группы, чтобы уложиться в промежутки *tz*). На основании этого ограничения формируется начальное построение групп партий и оптимизация их составов на втором уровне работы системы.

# ОПИСАНИЕ РАЗРАБОТАННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В ходе выполнения выпускной квалификационной работы бакалавра была поставлена задача на практике показать работоспособность модели системы и доказать необходимость методов оптимизации начальных решений на каждом уровне.

При выборе языка программирования были предложены несколько вариантов строго типизированных языков программирования с возможностью работы с динамическими объектами. Так же было необходимо учесть, что составляемая программа должна иметь оконный интерфейс. Для упрощения разработки были предложены языки программирования в составе систем разработки интерфейсов пользователя:

* С++ Builder;
* C++ Visual Studio;
* C# Visual Studio.

Среди перечисленных был выбран язык С#, так как этот язык является объектно-ориентированным (различные модули программы можно описать как отдельные классы) и в его работе присутствует система автоматической очистки памяти не использующихся переменных (что необходимо в работах модулей, основанных на методах, описанных выше).

## Описание разработанного модуля построения решения о составе партий данных фиксированного типа

В ходе выполнения выпускной квалификационной работы бакалавра была поставлена задача на практике показать работоспособность метода построения решения по составу партий фиксированного типа и доказать его необходимость в иерархической модели системы.

На основе рассматриваемого метода был построен модуль построения решения по составу партий данных фиксированного типа. В соответствии с описанным выше методом был написан алгоритм, формирующий решения по составу партий данных фиксированного типа. Данный алгоритм состоит из шагов:

1. Установка начальных значений.
2. Цикл while с условием *q1* <= *np1*.
3. Если не flag генерация нового решения.
4. Проверка нового решения в соответствии с теоремой 1 и удаление решения, если проверка не удалась.
5. Проверка возможности формирования нового решения (*h* + *j* < *mi*).
6. Изменение значения *h* если не выполняется условие теоремы 2.
7. Изменения значения *j* если выполняется условие теоремы 2.
8. Переход к следующему решения из матрицы *A1*.
9. Если сформировано хотя бы одно решение (*np2* > 0), то.
10. Удаление повторяющихся решений из матрицы *А2*.
11. Для каждого из *A2* происходит проверка текущего решения (изменение *i*-ой строки матрицы *А1* и передача её на второй уровень системы, получение решения со второго уровня и проверка градиента (буферизация индекса лучшего решения из матрицы *А2*)).
12. Если матрица *А2* не сформирована вернуть ложь.

Текущий алгоритм формирует решение и проверяет его на втором уровне для достижения общей цели. Проверка необходима, чтобы определить наилучший состав партий на текущем шаге алгоритма всей системы для конкретного типа данных. Далее дополнительными методами происходит возврат решения из модуля и модификация матрицы *А* (матрицы решения на текущем шаге алгоритма для всех типов данных). Описание методов возврата и дополнительных методов модуля были опущены, так как не являются глобальными методами всей системы в целом.

На рисунке 4.1 представлена структурная схема алгоритма формирования решения по составу партий данных, построенная на основании шагов, описанных выше.

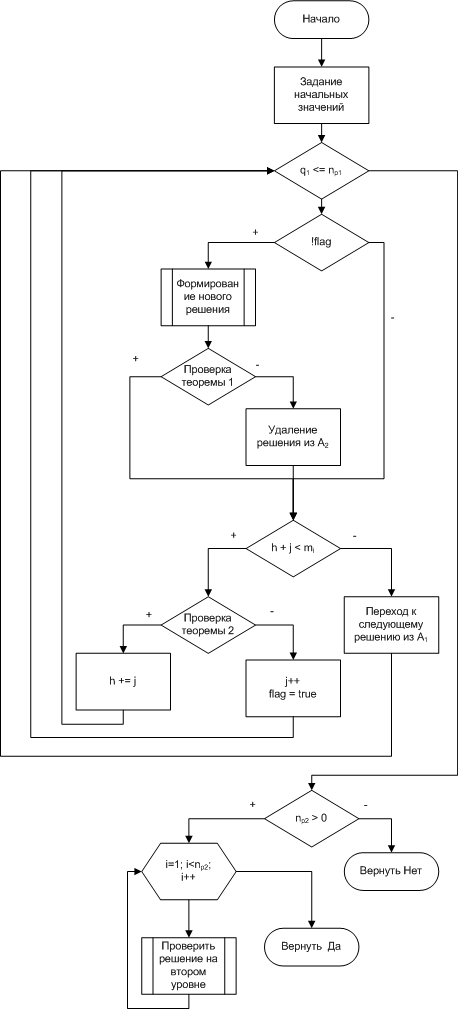


Рисунок 4.1 – Структурная схема алгоритма формирования решения по   
составу партий данных фиксированного типа

## Описание разработанного модуля построения решения о составах партий данных рассматриваемых типов

В ходе выполнения выпускной квалификационной работы бакалавра была поставлена задача на практике показать работоспособность модуля построения решения по составам партий рассматриваемых типов и доказать его необходимость при работе программы.

На основе рассматриваемого метода был построен модуль построения решения по составу партий данных фиксированного типа. В соответствии с описанным выше методом был написан алгоритм, формирующий решения по составу партий данных всех типов. Данный алгоритм состоит из шагов:

1. Установка начальных значений.
2. Формирование начального решения.
3. Отправка начального решения на второй уровень системы и вычисление критерия f1.
4. Буферизация критерия.
5. Если *I* – пустое множество, то выход.
6. Иначе буферизация списка рассматриваемых типов данных, матрицы решений *А* и установка флага решения значением «ложь»
7. Для всех рассматриваемых типов данных.
8. Формирование решения по составу партий данных текущего типа (отправка данных в модуль формирования решения по составам партий данных фиксированного типа и получение результирующей матрицы *А2* (матрицы всех составленных решений, полученных в ходе работы модуля)).
9. Проверка полученной на 8 шаге матрицы *А2*.
10. Если матрица пуста, то формируется решение второго типа для фиксированного типа данных (формируется начальное решение для фиксированного типа путем увеличения количества партий данных этого типа). На этом же шаге реализуется проверка полученного решения второго типа. Если решение не удовлетворяет условию теоремы 1, то этот тип не рассматривается в дальнейшем (текущий тип исключается из рассмотрения – удаляется из списка *I*).
11. Каждое решения из *А2* проверяется на втором уровне системы (измененная матрица *А* передается на второй уровень, возвращается результат и происходит расчет градиентов текущего решения). Если градиент меньше нуля, то текущее решение буферизируется как локальное оптимальное. После прохода по всех решениям матрицы *А2* для текущего типа данных сформировано локальное оптимальное решение по составу партий данных. Реализуется присвоение флагу решения значения «истина».
12. Проверка флага решения на равенство значению «истина».
13. Если проверка успешна, то текущее локальное оптимальное решение буферизируется как глобальное оптимальное решение. После прохода по всем типам данных на текущем шаге алгоритма всей системы (окрестность *Ок*) в буфере храниться максимальное решение среди всех рассматриваемых типов. Таким образом, сохраняется глобальное оптимальное решение по всем типам данных.
14. Реализуется переход на следующий шаг алгоритма (формирование окрестности большего размера *Ок+1*)

Текущий алгоритм формирует локальное оптимальное решение проходя по всем рассматриваемым типам данных на *Ок* шаге алгоритма функционирования всей системы. При рассмотрении всех типов данных на *Ок* шаге производится сохранения глобального оптимального решения и переход на шаг *Ок+*1.

При переходе на *Ок+*1 шаг реализуется увеличение окрестности решений (увеличение окрестности поиска глобального оптимального решения по составам партий всех типов данных).

На рисунке 4.2 представлена структурная схема метода формирования составов партий всех типов данных. На основании этой структурной схемы был написан программный модуль формирования составов партий данных на первом уровне работы программы.

Данный модуль использует в своей работе модуль оптимизации состава партий данных фиксированного типа, основанный на методе оптимизации состава партий данных фиксированного типа описанный выше и модуль второго уровня системы (модуль формирования групп партий данных).

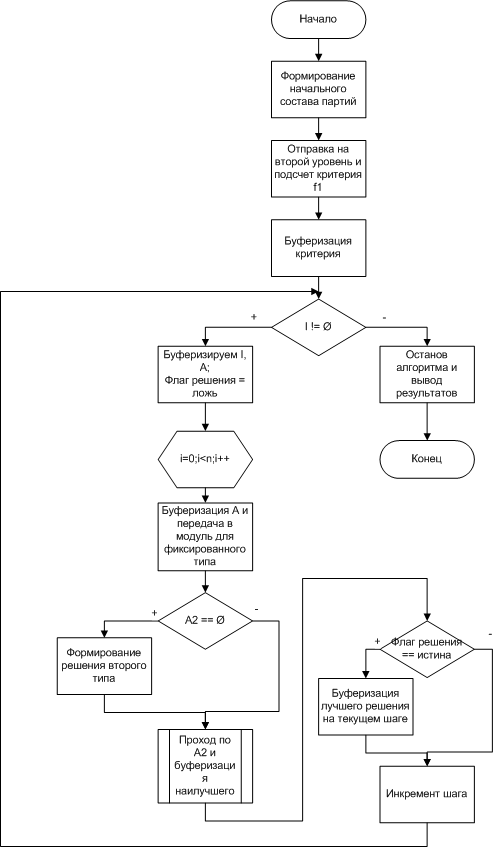


Рисунок 4.2 – Структурная схема метода формирования состава партий данных всех типов

## Выводы раздела 4

В данном разделе были рассмотрены и описаны программные модули функционирования системы, работающие на первом уровне. Описанные модули были построены на основании методов построения решения по составам партий данных, описанных в разделе 2. Каждый из модулей является независимым компонентом системы и решает свои задачи. В результате работы каждого из модулей формируется некоторое решение, которое рассматривается на нижестоящем уровне системы. Оптимальное глобальное решение по составам партий данных рассматриваемых типов и порядки их обработки мы получаем на первом уровне системы, так как передаем решение на нижестоящий уровень для получения групп и последующей оптимизации порядка обработки этих групп.

# АНАЛИЗ РЕЗУЛЬТАТОВ РАБОТЫ МЕТОДОВ ПОСТРОЕНИЯ РЕШЕНИЯ ПО ОБРАБОТКЕ ПАРТИЙ ДАННЫХ

Проведем анализ работы системы и докажет необходимость метода оптимизации состава партий данных на первом уровне системы. Сравнение необходимо проводить с аналогично построенной системой без модуля оптимизации составов партий данных на первом уровне.

Для сравнения возьмем фиксированные партии – используем начальное решение по составам партий данных всех типов. Фиксированные партии не будут модифицироваться, а сразу отправляться на второй уровень представленной модели (из фиксированных партий данных будут формироваться группы партий данных, проводится оптимизация расписания и проверка построенного расписания на временное ограничение )

В работе модуля оптимизации начальное решение улучшается (увеличивается окрестность решения на каждом последующем шаге алгоритма) при использовании метода оптимизации партий данных, описанного выше.

Проанализируем полученные данные.

Для анализа были взяты несколько промежутков временных ограничений работы системы и отношений максимальных к минимальным времен обработки и переналадки. В качестве временных промежутков были выбраны значение 60, 80 и 100 условных временных единиц. Так же введено ограничение на минимальное время обработки и переналадки системы (минимальное время обработки и переналадки равно 2 условным временным единицам).

Для всех временных промежутков установим фиксированное значение длины конвейера (количества обрабатывающих устройств в системе). Положим это значение равное 4 (в обработке всех требований используется 4 сегмента конвейера). Так же фиксированным является количество типов данных (n равно 4).

Проведем анализ системы при ограничении на время функционирования системы равное 60 условных временных единиц.

Таблица 5.1 – зависимость количества обработанных требований от отношения времени обработки и отношения времени переналадки партий требований при *tz* = 60.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста |  |  | Количество обработанных партий с методом оптимизации партий данных | Количество обработанных партий без метода оптимизации партий данных |
| 1 | 1 | 1 | 64 | 64 |
| 2 | 1 | 2 | 64 | 64 |
| 3 | 1 | 4 | 64 | 64 |
| 4 | 1 | 8 | 64 | 64 |
| 5 | 1 | 10 | 64 | 64 |
| 6 | 2 | 1 | 64 | 64 |
| 7 | 2 | 2 | 64 | 64 |
| 8 | 2 | 4 | 60 | 50 |
| 9 | 2 | 8 | 50 | 36 |
| 10 | 2 | 10 | 48 | 36 |
| 11 | 4 | 1 | 40 | 8 |
| 12 | 4 | 2 | 36 | 8 |
| 13 | 4 | 4 | 30 | 8 |
| 14 | 4 | 8 | 24 | 8 |
| 15 | 4 | 10 | 20 | 8 |
| 16 | 8 | 1 | 20 | 8 |
| 17 | 8 | 2 | 16 | 8 |
| 18 | 8 | 4 | 12 | 8 |
| 19 | 8 | 8 | 12 | 6 |
| 20 | 8 | 10 | 10 | 6 |
| 21 | 10 | 1 | 10 | 4 |
| 22 | 10 | 2 | 10 | 4 |
| 23 | 10 | 4 | 10 | 4 |
| 24 | 10 | 8 | 10 | 4 |
| 25 | 10 | 10 | 10 | 2 |

Рисунок 5.1 – график зависимости количества обработанных данных от номера теста при *tz* = 60

Таблица 5.2 – зависимость количества обработанных требований от отношения времени обработки и отношения времени переналадки партий требований при *tz* = 80.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста |  |  | Количество обработанных партий с методом оптимизации партий данных | Количество обработанных партий без метода оптимизации партий данных |
| 1 | 1 | 1 | 64 | 64 |
| 2 | 1 | 2 | 64 | 64 |
| 3 | 1 | 4 | 64 | 64 |
| 4 | 1 | 8 | 64 | 64 |
| 5 | 1 | 10 | 64 | 64 |
| 6 | 2 | 1 | 64 | 64 |
| 7 | 2 | 2 | 64 | 64 |
| 8 | 2 | 4 | 64 | 64 |
| 9 | 2 | 8 | 64 | 64 |
| 10 | 2 | 10 | 64 | 50 |
| 11 | 4 | 1 | 50 | 8 |
| 12 | 4 | 2 | 40 | 8 |
| 13 | 4 | 4 | 36 | 8 |
| 14 | 4 | 8 | 36 | 8 |
| 15 | 4 | 10 | 30 | 8 |
| 16 | 8 | 1 | 30 | 8 |
| 17 | 8 | 2 | 24 | 8 |
| 18 | 8 | 4 | 20 | 8 |
| 19 | 8 | 8 | 16 | 8 |
| 20 | 8 | 10 | 12 | 8 |
| 21 | 10 | 1 | 12 | 8 |
| 22 | 10 | 2 | 12 | 8 |
| 23 | 10 | 4 | 12 | 8 |
| 24 | 10 | 8 | 12 | 6 |
| 25 | 10 | 10 | 10 | 4 |

Рисунок 5.2 – график зависимости количества обработанных данных от номера теста при *tz* =80

Таблица 5.3 – зависимость количества обработанных требований от отношения времени обработки и отношения времени переналадки партий требований при *tz* = 100.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста |  |  | Количество обработанных партий с методом оптимизации партий данных | Количество обработанных партий без метода оптимизации партий данных |
| 1 | 1 | 1 | 64 |  |
| 2 | 1 | 2 | 64 |  |
| 3 | 1 | 4 | 64 |  |
| 4 | 1 | 8 | 64 |  |
| 5 | 1 | 10 | 64 |  |
| 6 | 2 | 1 | 64 |  |
| 7 | 2 | 2 | 64 |  |
| 8 | 2 | 4 | 64 |  |
| 9 | 2 | 8 | 64 |  |
| 10 | 2 | 10 | 64 |  |
| 11 | 4 | 1 | 50 |  |
| 12 | 4 | 2 | 50 |  |
| 13 | 4 | 4 | 50 |  |
| 14 | 4 | 8 | 50 |  |
| 15 | 4 | 10 | 48 |  |
| 16 | 8 | 1 | 36 |  |
| 17 | 8 | 2 | 30 |  |
| 18 | 8 | 4 | 26 |  |
| 19 | 8 | 8 | 24 |  |
| 20 | 8 | 10 | 22 |  |
| 21 | 10 | 1 | 20 |  |
| 22 | 10 | 2 | 20 |  |
| 23 | 10 | 4 | 18 |  |
| 24 | 10 | 8 | 16 |  |
| 25 | 10 | 10 | 12 |  |

Рисунок 5.3 – график зависимости количества обработанных данных от номера теста при *tz* = 100

На основании полученных данных можно смело заявить, что при использовании методов оптимизации на первом уровне системы мы получаем лучшее решение, чем решение, основанное на фиксированных партиях. Алгоритм оптимизации получает оптимальное решение по составу групп партий данных при ограничении на время обработки .

## Выводы раздела 5

В данном разделе были получены наглядные результаты работы системы, а так же проведен сравнительный анализ и доказательство необходимости использования методов оптимизации решений на первом уровне по отношению к фиксированным партиям данных.

# ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА

## Маркетинговые исследования программного продукта

### Сущность и этапы маркетинговых исследований

При разработке нового проекта были проведены маркетинговые исследования программного продукта.

Маркетинговые исследования - систематический и объективный сбор и анализ информации, связанной со сбытом продукции и предложением услуг. Проведение исследований увеличивает вероятность применения наиболее эффективных маркетинговых действий. Часто исследования предпринимают как реакцию на существующую проблему, хотя маркетинговая информация может быть использована и как основа для перспективного планирования [15].

Основная цель маркетинговых исследований ‑ уменьшить неопределенность и минимизировать риск в процессе принятия управленческих решений; следить за процессом реализации маркетинговых задач.

Этапы маркетинговых исследований приведены в приложении Б [15]

### Оценка рыночной направленности программного продукта

Для оценки рыночной направленности программного продукта проведем исследование ПП с помощью построения экспертной таблицы Б.1 (приложение Б).

Найдем среднее арифметическое полученных оценок:



При таком значении полученного среднего балла разрабатываемый программный продукт обладает рыночной направленностью.

### Конкурентоспособность продукта

Текущие системы построения расписаний имеют большее количество ограничений при работе, а также затрачивают большее время на работу. Разрабатываемый программный продукт основан на новом алгоритме построения расписаний и является программным подтверждением эффективности разработанного нового алгоритма построения расписаний. Исходя из этого можно сделать вывод, что программный продукт имеет конкурентное преимущество.

В дальнейшем подобные алгоритмы построения расписаний будут использоваться в коммерческих целях на крупных производственных предприятиях и нагруженных серверах, а так же применяться в космической промышлености при обработке большого количества снимков со спутников.

### Сегментация рынка

Сегментация рынка – процесс разбивки потребителей или потенциальных потребителей на рынке на различные группы (или сегменты), в рамках которых потребители имеют схожие или аналогичные запросы, удовлетворяемые определенным комплексом маркетинга.

Сегментация потребительского рынка может быть произведена по нескольким признакам:

Сегментирование по географическому признаку включает разделение рынка на различные географические единицы (переменные): регион, область, район, размер города, плотность.

Сегментирование по демографическому признаку заключается в разделении рынка в соответствии с такими переменными как: возраст, пол, размер семьи, жизненный цикл семьи род занятий, уровень дохода, образование, национальность, вероисповедание.

Сегментирование по поведенческому признаку заключается в выделении групп покупателей на основе их знаний, квалификаций как пользователей и их реакций на товар [16].

### Предполагаемый потребитель

Проанализировав рынок, были получены сведения, что разработанная система подходит для конвейерного производства различных деталей на одной линии производства. Список всех подходящих производств представлен в приложении Б в таблице Б.2.

### Жизненный цикл программного продукта

Жизненный цикл ПП – время существования товара на рынке.

Жизненный цикл ПП включает в себя этапы:

1. Исследование и разработка. На этом этапе происходит зарождение товара, его идеи. Сбыт товара пока еще нулевой, прибыть отрицательна.

2. Внедрение. На этом этапе товар начинает свое продвижение к потребителю, происходит активная рекламная кампания, но при росте сбыта прибыль продолжает расти в отрицательную сторону.

3. Этап роста. Самый благоприятный этап для производителя. Предприятие получает значительную прибыль, сбыт товара продолжает расти.

4. Этап зрелости. Товар выпускается большими партиями, сбыт растет уже не такими высокими темпами, прибыль понемногу снижается, так как чувствуется конкуренция.

5. Этап спада. Сбыт резко падает, предприятие перестает выпускать товар, прибыль очень низка [17].

Схема этапов жизненного цикла представлен в приложении Б (рисунок Б.1).

### Уровни программного продукта как товара

Программный продукт представляет собой товар, а любой товар начинается с идеи. Создание нового продукта связано с возникновением инновационной идеи, то есть освоение нового рынка, расширение или появление новых способов производства, возникновением или расширением потребностей потребителя, появлением новых функций продукта.

1. Товар по замыслу, товар по сути, сущность продукта: товар есть способ удовлетворения определенной потребности человека.

2. Товар в реальном исполнении, фактический продукт: товар есть предмет материального мира.

3. Товар с расширением, дополнительный продукт: дополнительные выгоды, которые может получить потребитель сверх фактического продукта [18].

Рисунок трехуровневой модели товара представлен в приложении Б (рисунок Б.2).

### Итоги маркетингового исследования

В данном разделе были произведены маркетинговые исследования программного продукта, произведена оценка потенциальных потребителей.

Проведено сегментирование рынка, в результате чего был выделен сегмент, на который следует ориентировать ПП. Кроме того был определен жизненный цикл программного продукта.

## Определение затрат на проектирование продукта

### Расчет трудоемкости программного продукта

Трудоемкость проекта определяется, исходя из данных об используемых функциях ПП. Определим эти функции, которые приведены в таблице Б.3 (приложение Б).

Общий объем разрабатываемого ПП (*V*0) определяется в тысячах условных машинных команд по формуле:

где  – объем i – ой функции ПП, тыс. УМК;

*n* – общее число функций ПП;

,

где T0-общая трудоемкость.

Трудоемкость считается путем суммирования затрат труда по всем операциям, в данном случае, при разработке ПП (Расчеты произведены и помещены в приложение Б таблица Б.4, значение ).

Исходя из полученной трудоемкости и численности исполнителей, можем рассчитать срок разработки ПП:

,

,

где *Ф* – среднее количестводней в месяце, равное 20,56 дней,

*Ч* – численность разработчиков ПП,

*Чр* –число рабочих часов в сутках, равное 8.

Рассчитаем эффективный фонд времени – разница между номинальным фондом времени (*F*ном) и потерями рабочего времени (*П*).

*F*ном принимаем равным 1974;

Для ЭВМ:

### Расчет эксплуатационных затрат разработчика

К эксплуатационным затратам относятся затраты, связанные с обеспечением нормального функционирования проекта. Данные для расчета годовых эксплуатационных затрат представлены в таблице Б.5 (приложение Б).

Для того, чтобы определить сумму годовых эксплуатационных затрат необходимо выполнить следующие расчеты. Расчеты приведены в таблице Б.6 (приложение Б)

Рассчитаем себестоимость часа машинного времени (*Сч.м.в*) по формуле:

,

где - сумма годовых эксплуатационных затрат, руб.

,

### Расчет сметы затрат на проектирование

Смета затрат - это перечень видов затрат которые включенных в себестоимость общего объема выпускаемой продукции, представляемых услуг. Стоимость всех работ, выполняемых при разработке ПП, можно разделить на две части:

Стоимость работ по разработке и отладке программного обеспечения, выполняемых с помощью вычислительной техники, срок разработки равен 1 месяцу (*Срм*= 1);

Стоимость работ, производимых без применения вычислительной техники, срок разработки равен 0,44 месяца (*Срр*=0,44).

Расчеты приведены в таблице Б.7 (приложение Б)

## Расчет капитальных затрат

Капитальные вложения для разработчика – расходы на покупку (*Цтс*), доставку (*Зтр*) и монтаж (*Зм*) технических средств, а также приобретение программного обеспечения (*Цоб*),необходимого для процесса создания программного продукта:

,

где *Цтс* – расходы на покупку, цена = 21000 (руб.);

*Зтр* – 5% транспортные затраты от стоимости оборудования.   
*Зтр* = 21000 *×* 5 / 100= 1050 (руб.);

*Зм* – затраты на монтаж технических средств 8% от прейскурантной цены. *Зм* = 21000 *×* 8 / 100 = 1680 (руб.);

*Цпо* – затраты на приобретение программного обеспечения (лицензия) составляют 2500 (руб.).

*Кр* = 21000 + 1050 + 1680 + 2500 = 26230 (руб.).

## Формирование цены предложения разработчика

Формирование цены разработчика производится методом безубыточности.

Подробные расчеты формирования цены приведены в таблице Б.8 (приложение Б)

Рисунок 4.3 – График безубыточности

Точка безубыточности, которая показывает, что объем минимальных продаж составляет 6, а максимальных 10 ПП.

## Оценка экономической эффективности программного продукта

Экономическая эффективность позволяет судить о необходимости внедрения программного продукта. Факторы, влияющие на эффективность разработки, подразделяются на две большие группы: внутренние и внешние. Внутренние факторы эффективности проекта создания программного продукта подразделяются на структурные, ресурсные, технические, управленческие, рыночные, качество интеллектуального капитала.

В основе оценки эффективности ПП лежит его цена – чем выше цена продукта, тем более вероятна его экономическая эффективность. В качестве нижнего предела цены принимаются предельные или полные затраты производителя на разработку, тиражирование и сопровождение программного продукта. В качестве верхнего предела цены выбирается рыночная цена с ориентацией на конкурента и рыночную конъюнктуру. Порядок расчета показателей экономической эффективности приведен в приложении В.

Результаты расчета сводятся в таблицу Б.9 (приложение Б)

Рисунок 4.4 – График финансового профиля проекта

В таблице 6.1 приведены все показатели, полученные при анализе графика финансового профиля проекта.

Таблица 6.1 – Показатели эффективности проекта

|  |  |  |
| --- | --- | --- |
| Показатели | Единица измерения | Величина |
| Прибыль проекта | руб. | 12240,64 |
| Интегральный экономический эффект | руб. | 30660,16 |
| Рентабельность проекта | % | 47,6 |
| Срок окупаемости | год | 0,6 |
| Период возврата капиталовложений | год | 1,6 |

## Выводы раздела 6

В ходе выполнения экономического исследования проекта было произведено маркетинговое исследования разработанного программного продукта, а также рассчитаны показатели, определяющие целесообразность, направленность и рентабельность проекта. Затраты на проектирование составили 27865,03 руб. Данный показатель является неотъемлемой составляющей формирования цены ПП, оказывающий существенное влияние на её окончательную величину. В результате была получена следующая цена за единицу программного продукта 7200,38 руб. Данный проект разработки и внедрения программного продукта является целесообразным, так как получен положительный экономический эффект от разработки ПП и прибыль составила12240,64 руб.

# БЕЗОПАСНОСТЬ ЖИЗНИДЕЯТЕЛЬНОСТИ

## Основные сведения

Охрана труда является важной составляющей деятельности любого предприятия. Она направлена на обеспечение эффективного, безопасного и комфортного труда работника.

В нашем расчетно-графическом задании у нас есть производственное помещение со следующими характеристиками:

* габариты помещения, м: 6,5 × 5,5 × 3;
* ориентация наружной стены здания: ЮЗ;
* площадь остекления, м: 4,8;
* средства солнцезащиты: светлые жалюзи;
* тип и мощность ламп системы освещения, кВт: люминесцентные,0,06;
* количество ламп системы освещения: 16;
* число рабочих мест: 6.

Задачей раздела БЖД является анализ условий труда и разработка мер по его улучшению в заданном помещении по следующим критериям:

* планировка оборудования и мест;
* тяжесть и напряженность труда;
* качество воздуха и микроклимат рабочей зоны;
* шум и вибрация;
* освещение;
* электро- и пожаробезопасность.

## Краткая характеристика помещения и выполняемых работ

Рабочее помещение имеет размеры 6,5 × 5,5 × 3 (м). Число работников данного помещения составляет 6 человек.

Площадь помещения 35,75 (м2), объем помещения 107,25 (м3). На одного работника приходится примерно 6 (м2) по площади и 17,9 (м3) по объему.

В помещении расположено 6 рабочих мест, шкаф, две кадки с растениями, мультимедийная доска и огнетушитель. Рабочие места, за исключением места начальника, отделены друг от друга перегородками высотой 1,3 (м) в связи тем, что работа требует умственного напряжения и концентрации внимания.

На каждом рабочем месте расположено по комплекту вычислительной техники, канцелярские принадлежности. На рабочем месте начальника отдела расположен также матричный принтер.

В шкафу располагается документация отдела, в том числе по охране труда и пожарной безопасности, также имеется аптечка, включающая следующие средства:

* обезболивающие и противовоспалительные средства: диклофенак, пакет-контейнер портативный гипотермический (охлаждающий), капли глазные «Искусственная слеза»;
* средства для обработки кровотечения, перевязки и обработки ран: жгут, индивидуальный стерильный перевязочный пакет, бинт стерильный, бинт нестерильный, лейкопластырь, медицинские перчатки, перекись водорода, йод, зеленка, вата;
* средства при болях в сердце: нитроглицерин, валидол;
* средства для сердечно-легочной реанимации и клинической смерти: устройство для проведения искусственного дыхания;
* средства при обмороке: нашатырный спирт;
* средства при аллергиях и отравлениях: активированный уголь, супрастин, тавегил;
* средства при стрессовых реакциях: корвалол или настойка валерианы, каптопресс;
* ножницы с тупыми концами;
* блокнот и ручка.

Комплектность аптечки постоянно контролируется, лекарственные и другие средства постоянно пополняются или заменяются по мере истечения срока годности.

Ориентация натужной стены здания является юго-западной, площадь остекления составляет 4,8 (м2). Тип и мощность ламп системы освещения –люминесцентные мощностью 0,06 (кВт), количество ламп системы освещения составляет 16 штук.

В помещении используется офисная техника:

* ПЭВМ с житкокристаличным дисплеем;
* Эргономичная клавиатура и мышь;
* Матричный принтер.

Виды выполняемых работ на ПЭВМ:

* программирование программных продуктов под заказ;
* осуществление модернизации и поддержки ПО.

## Планировка и размещение оборудования и рабочих мест

На рисунке Г.1 (приложение Г) представлена схема помещения, на которой показано размещение рабочих мест и оборудования с указанием размеров, дверных и оконных проемов, проходов, офисной мебели и средств пожаротушения.

Цифрами на эскизе обозначены: 1 – огнетушитель; 2 – шкаф (для хранения аптечки, документации, в том числе по охране труда и пожарной безопасности);   
3 – рабочее место (рабочий стол с установленным на нем ПК и стул); 4 ‑ рабочее место начальника отдела (рабочий стол с ПК и матричным принтером, а также стул); 5 – сетевой фильтр на 6 розеток; 6 – кадка с растением; 7 – сетевой рубильник-автомат; 8 – мультимедийная доска.

Планировка соответствует нормативным требованиям СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организация работы» [19], за исключением того, что рабочие места не удалены на расстояние 1 метра от стен, а так же боковые поверхности мониторов расстояние не удалены на расстояние 1.2 метра. Ко всем рабочим местам обеспечивается свободный проход. Используются ЖК мониторы при работе с ПЭВМ. Площадь на каждое рабочее место в пределах нормы.

## Тяжесть и напряженность труда. режим труда и отдыха

Далее приведена оценка тяжести и напряженности работы. Данные сведены в таблицу Г.1 (приложение Г):

Таким образом, класс условий труда – 2.

В работе предусмотрены технологические перерывы по 10 минут каждый час, что суммарно составляет 80-90 минут за смену. Данное значение соответствует нормам для II группы работы с ПЭВМ при 8-часовой смене.

В течении технологического перерыва выполняются упражнения для успокоения и снятия зрительного напряжения у сотрудников.

## Качество воздуха и микроклимат рабочей зоны

Работа программиста относится к категории легких работ. Нормами микроклимата по данной приведены в таблице Г.2 (приложение Г) [20].

В нашем случае работа относиться к категории Ia. В течении года должна обеспечиваться температура 22-25 ˚С. Для холодного периода в помещении есть система отопления, обеспечивающая требуемый температурный режим. Для теплого периода наблюдаются превышения допустимого температурного режима, в связи с чем рекомендуется установить систему кондиционирования воздуха, расчет и выбор кондиционера рассмотрен в пункте 7.11.

Контакт со специфичными при работе с ПЭВМ веществами (тонер, озон и т.д.) не представляет опасности, т.к. имеется вентиляция. Источники выделения вредных веществ отсутствуют.

Далее приведена информация о составе воздуха на рабочих местах, сведенная в таблицу Г.3 (приложение Г).

Качество воздуха находится в пределах норм. Вещества, состоящие в воздухе, не превышают допустимых норм ПДК.

## Шум и вибрация

Для оценки шума используют частотный спектр измеряемого уровня звукового давления, выраженного в дБ, который сравнивают с предельным спектром, приведены в таблице Г.4 (приложение Г) [21].

Уровень шумов от ЭВМ и другого оборудования, используемого при разработке, незначительный. В данном случае его уровень определяется только хозяйственной деятельностью человека и составляет 35 (дБ), что соответствует нормам.

## Освещение

Естественное освещение имеет доступ в помещение через три окна расположенных на одной из стен. Общая площадь оконных проемов 4,8 (м2). Окна имеют южную ориентацию. На них установлены светлые жалюзи в качестве солнцезащиты.

Возле окон распложено 3 рабочих места. Так как имеются средства солнцезащиты, то при попадании солнечного света в окна работники не имеют негативного воздействия.

В помещении установлено 16 потолочных люминесцентных ламп. Линии источников света (ламп) располагаются над рабочими местами (рисунок Г.2 приложение Г).

Расстояние между светильниками в ряду составит l1=(7,5 × 1,255 × 2)/3=0,9 (м)

Расстояние между рядами l2=5,5/4=1,375 (м) а между одним рядом 2,75/2=1,375 (м). Удаление ряда светильников от стены 1,375 (м). Что не превышает рекомендованных 1,4 × 2,14=3 (м) и 0,7 × 2,14=1,498 (м).

В таблице Г.5 (приложение Г) представлены нормы освещенности рабочего места и фактические значения, измеренные в помещении

Освещение удовлетворяет требуемым нормам.

## Электро - и пожаробезопасность

Помещение по опасности поражения электрическим током отнесем к категории без повышенной опасности, т.к. данное помещение характеризуется отсутствием условий, которые создают особую или повышенную опасность [22].

Помещение по категории пожарной опасности относятся к B1-B4 (пожароопасная).

В таблице Г.6 (приложение Г) сведены данные о возможных классах пожаров.

Для обеспечения пожаробезопасности помещения необходим один порошковый огнетушитель с вместимостью огнетушащего вещества 10/9 (кг) и покрывающим 200 (м2).

Для обеспечения электробезопасности используется:

Сетевой рубильник-автомат;

Кабельные трассы в скрытых коробах.

Организационно-технические мероприятия по обеспечению электро- и пожаробезопасности:

* проведение инструктажей, наличие инструкций по ОТ, а также плана эвакуации при пожаре;
* наличие в помещении средств пожаротушения;
* соблюдение правил пользования ПЭВМ.

Данное помещение полностью соответствует правилам и нормам охраны труда по электро- и пожаробезопасности.

## Статическое электричество и излучения

Компьютер и вспомогательное электрооборудование формирует сложную электромагнитную обстановку на рабочем месте.

Установлено, что ЭМП негативно влияют на центральную нервную систему, вызывая головные боли, головокружения, тошноту, депрессию, бессонницу, отсутствие аппетита, возникновение синдрома стресса.

Поэтому проанализируем способы защиты от статического электричества и излучений в помещениях с ПЭВМ.

К основным мерам профилактики неблагоприятного влияния электромагнитного поля относятся:

* своевременное прохождение периодических медосмотров;
* отключение оборудования, на котором временно не работают, но находятся рядом;
* использование мониторов, соответствующих современным требованиям по защите от излучений, предпочтение следует отдавать жидкокристаллическим мониторам;
* выполнение режима труда и отдыха, не превышение общего времени взаимодействия с ПЭВМ за смену.

Для снижения влияние электростатического поля необходимо:

* устанавливать нейтрализаторы статического электричества;
* поддерживать в помещении относительную влажность не ниже 45-50% и ежедневно проводить влажную уборку;
* для снятия заряда несколько раз в день мыть руки и лицо водой;
* протирать экран и рабочее место специальной антистатической салфеткой;
* ограничить количество полимерных материалов в помещении.

## Эргономика и техническая эстетика

Проанализировав рабочие места в соответствии с СанПиН 2.2.2/2.4.1340-03 «Система стандартов эргономики и технической эстетики. Основные положения», была заполнена таблица Г.7 (приложение Г) по параметрам рабочей поверхности и приведен эскиз рабочего места и рабочей поверхности.

В результате анализа было выяснено, что параметры рабочего места не имеют диапазон регулирования, соответствующего СанПиН 2.2.2/2.4.1340-03 [23].

Рациональное цветовое оформление помещения направленно на улучшение санитарно-гигиенических условий труда, повышение его производительности и безопасности. Окраска помещений ВЦ влияет на нервную систему человека, его настроение и в конечном счете на производительность труда. Основные производственные помещения целесообразно окрашивать в соответствии с цветом технических средств. Учитывая ориентацию окон помещения можно сделать вывод, что лучше всего подойдет цвет красно-оранжевый для пола и бирюзовый(зелено-голубой) для стен.

## Расчет искусственного освещения производственных помещений

Помещение имеет следующие характеристики длина А=6,5 (м), ширина В=5,5 (м), высота потолка 3 (м), коэффициенты отражения потолка - 70%, стен -50%, пола -30%, высота рабочих поверхностей столов 0,8 (м) [18].

Свес светильника можно принять ,где Н- габаритный размер светильника.

Для определения коэффициента использования светового потока рассчитаем индекс помещения по формуле:

*,*

где А, В – длина и ширина помещения в м.

Рассчитаем количество светильников по формуле:

*,*

где *En* – нормируемая минимальная освещенность,лк (300 лк)

*Кз* – коэффициент запаса (1,4)

*S* – площадь освещаемого освещения

*Z* – коэффициент неравномерности помещения (1,1)

*Fл* – номинальный световой поток одной лампы, лм (по таблице 1100)

*N* – число ламп в светильнике

*η* – коэффициент использования светового потока в долях единицы (по таблице *η* = 0.58)

*h*-расчетная высота подвесов светильников, м.

Рассчитаем необходимое для рассматриваемого помещения количество светильников ЛПО 36-4\*20 УХЛ4.

Принимаем *N*=6.Тогда мощность системы освещения составит   
*P*=6 × 4 × 20=480 (Вт).

Учитывая требования энергоснабжения, рассмотрим возможность применения и помещения светильников ЛСО 05-2\*40-001 УХЛ4. Необходимое число светильников такого типа составит:

Принимаем *N*=4.Таким образом, мощность системы освещения при использовании светильников ЛСО 05-2\*40-001 УХЛ4 составит *Р*=2 × 2 × 40=160 (Вт), что на 33% ниже чем в первом случае. Светильники размещаем в 2 ряда, параллельных стене со светопроемами. Такое расположение позволяет рационально использовать искусственное освещение в зависимости от естественного.

Расстояние между светильниками в ряду составит l1=(6,5×1,255×2)/3=0,9 (м).

Расстояние между рядами l2=5,5/2=2,75 (м) а между одним рядом 2,75/2=1,375 (м). Удаление ряда светильников от стены 1,375 (м). Что не превышает рекомендованных 1,4 × 2,14=3 (м) и 0,7 × 2,14=1,498 (м).

Таким образом установка 4 светильников ЛСО 05-2\*40-001 УХЛ4 позволит выполнить требования к равномерности и уровню освещенности рабочих поверхностей при снижении энергопотребления на 33% по сравнению с 6 светильниками ЛПО 36-4\*20 УХЛ4.

## Выводы раздела 7

В данной работе был выполнен анализ помещения на удовлетворение нормам по различным показателям:

* планировка и размещение оборудования и рабочих мест;
* тяжесть и напряженность труда;
* качество воздуха и микроклимат рабочей зоны;
* шум и вибрация;
* освещение;
* электро- и пожаробезопасность;
* статическое электричество и излучение;
* эргономика и техническая эстетика.
* В ходе анализа было выявлено несколько недочетов:
* планировка рабочих мест не удовлетворяет требованию о расстоянии от стен не менее метра, а так же расстояние между боковыми поверхностями мониторов не менее 1,2 метра;
* был произведен расчет искусственного освещения производственных помещений для проверки на соответствие санитарным нормам производственного помещения.

# ЗАКЛЮЧЕНИЕ

В результате выполнения дипломной работы реализован и исследован метод построения расписаний обработки данных разных типов в конвейеризированной системе. Так как выпускная работа является комплексной, то в рамках данной пояснительной записки рассматриваются только верхний и нижний уровни иерархии системы.

В данной работе рассматриваются некоторые алгоритмы формирования эффективных расписаний обработки данных, однако, не один из них не может быть использован для построения расписаний групповой обработки.

Задача составления расписания является трудновыполнимой, поэтому в данной работе предлагается провести вертикальную декомпозицию обобщенной цели системы и разбить ее на совокупность иерархически упорядоченных уровней (подцелей). При этом достижение каждой из подцелей ведет к выполнению обобщенной цели системы. Особенностью данной задачи является:

* задаваемые типы данных;
* задаваемое количество данных каждого определенного типа, что позволяет перейти к формированию партий данных;
* задаваемой ограничение на длительность реализации.

В ходе работы была описана и обоснована трехуровневая иерархическая теоретико-игровая модель, которая реализует формирование эффективных расписаний групповой обработки партий данных. Обобщенной целью системы является обработка максимального количества требований (минимизация необработанных требований). Каждый уровень модели выполняет следующие действия:

* первый уровень модели формирует эффективный состав партий требований. Критерием оптимальности данного уровня является количество необработанных требований. Сформированный состав партий передается на второй уровень;
* второй уровень формирует эффективный состав групп партий и пул необработанных данных. Для данного уровня было введено два критерия оптимальности: критерий, реализующий внутреннюю цель (критерий 1), и внешний, реализующий внешнюю цель (критерий 2). Алгоритм имеет возможность выполнять поиск эффективного решения по одному из критериев (второй при этом так же вычисляется и фиксируется). Сформированные группы передаются на третий уровень. Эффективное решение (эффективный состав групп партий) передается на верхний уровень;
* третий уровень формирует эффективное расписание обработки групп.

Для получения оптимального решения вышестоящий уровень системы использует решение, полученное на нижестоящем уровне системы. Для оценки эффективности решения на каждом из уровней используются критерии оценки, которые учитывают:

* на первом (верхнем) уровне – общее количество требований, обработанных в системе в течение *Z* интервалов времени;
* на втором (среднем) уровне – общую эффективность использования оборудования конвейерной системы при обработке всех партий требований группы;
* на третьем (нижнем) уровне – эффективность использования оборудования конвейерной системы при размещении рассматриваемой партии (добавляемой в расписание для соответствующей группы) в последовательностях ;

На основе теоретико-игровой модели, а так же алгоритмов формирования партий, групп и расписаний было реализовано приложение средствами языка С#. Интерфейс данного приложения позволяет задавать все необходимые параметры системы, а на выходе получать эффективное решение для всех трех уровней иерархии. С помощью этого приложения были проведены исследования эффективности алгоритма формирования эффективного состава групп партий.

В разделе «Экономическое обоснование проектных решений» было проведено экономическое обоснование проектирования и разработки описанного программного продукта. Так же было проведено маркетинговое исследование и составлен список потенциальных покупателей программного продукта.

В разделе «Охрана труда» был проведен анализ параметров производственного помещения и разработана оптимальная планировка для него.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Топорков, В.В. Модели распределенных вычислений [Текст] / В.В. Топорков. – М.: Физматлит, 2004. – 320 с.
2. Ковалёв, М.Я. Модели и методы календарного планирования [Текст] : курс лекций / М.Я. Ковалёв. – Минск: БГУ, 2004. – 63 с.
3. Танаев, В.С. Теория расписаний. Групповые технологии [Текст] / В.С. Танаев, М.Я. Ковалёв, Я.М. Шафранский. – Минск: ИТК НАН Беларуси, 1998. – 290 с.
4. Лазарев А. А., Гафаров Е. Р. Теория расписаний. Задачи и алгоритмы. М.: МГУ, 2011. ‑ 222 с.
5. Лазарев А. А. Мусатова Е. Г., Кварцхелия А. Г., Гафаров Е. Р. Теория расписаний. Задачи управления транспортными системами. М.: МГУ им. М. В. Ломоносова, 2012, ‑ 159 с.
6. Ковалев М. М. Модели и методы календарного планирования. Курс лекций. Минск: БГУ, 2004. ‑ 63 с.
7. Crauwels, H.A.J., Potts, C.N., Van Oudheusden, D. and Van Wassenhove, L.N. (2005) Branch and bound algorithms for single machine scheduling with batching to minimize the number of late jobs. Journal of Scheduling, 8, (2), p. 161-177
8. Maksim S. Barketau, T.C. Edwin Cheng, Mikhail Y. Kovalyov, C.T. Daniel Ng. (2007) Batch Scheduling of Deteriorating Products Decision Making in Manufacturing and Services 1, p. 25-34.
9. Месарович, М. Теория иерархических многоуровневых систем [Текст] / М. Месарович, Д. Мако, И. Такахара – М.: Наука, 1973. – 340 с.
10. Петросян, Л.А. Теория игр. [Текст]: Учебное пособие / Л.А. Петросян, Н.А. Зенкевич, Е.А. Семина. – М.: Высшая школа, 1998. – 304 с.
11. Новиков Д.А. Теория управления организационными системами [Текст] / Д. А. Новиков. – М.: МПСИ, 2005. – 584 с.
12. Шеллинг Т. Стратегии конфликта [Текст] / Томас Шеллинг : пер. с англ. Т. Даниловой – М.: ИРИСЭН, 2007. – 376 с.
13. Гермейер, Ю.Б. Игры с непротивоположными интересами [Текст] / Ю.Б. Гермейер. – М. Наука, 1976. – 328 с.
14. Кротов, К.В. Градиентный метод составления расписаний в многостадийной системе с различными временами поступления данных [Текст]. / К.В. Кротов // Вісник СевНТУ «Автоматизация процесів та управління»: збірник наукових праць. – 2010 – Вип. 108. – 145-151 c.
15. Баркен Д.И. Маркетинг для всех / Д.И. Баркен -М.: Редакционно-издательский центр «Культ-информ-пресс», 1991г. – 257с.
16. Попов Е. В. Продвижение товара. Екатеринбург: Наука. 1997. ‑ 350 с.
17. Попов Е.В., Попова Л.Н, Клюев Ю.Б. Разработка товара. Екатеринбург: Изд-во Урал.гос. техн. ун-та, 1997. – 58 c.
18. Дорошев В. И. Товар в маркетинговой деятельности предприятия: Учеб.пособие. / С. - Петерб. торг. - экон. ин-т. - СПб, 1996. ‑ 62 с.
19. «Гигиенические требования к персональным электронно-вычислительным машинам и организация работы». СанПиН 2.2.2/2.4.1340-03.
20. «Гигиенические требования к микроклимату производственных помещений». СанПиН 2.2.4.548-96.
21. «Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории жилой застройки». СанПиН 2.2.4/2.1.8.562-96.
22. СП 1.13130.2009 «Системы противопожарной защиты. Эвакуационные пути и выходы».
23. «Гигиенические требования к естественному, искусственному и совмещенному освещению жилых и общественных зданий». СанПиН 2.2.1/ 2.1.1.1278-03; ДЕАН - Москва, 2003. - **876** c.

# ПРИЛОЖЕНИЕ А

**Код программных модулей**

Код модуля построения групп партий всех типов данных:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Diplom

{

class FirstLevel

{

private List<int> I;//Вектор интерпритируемых типов данных

private List<int> Ii;//Вектор интерпритируемых типов данных на текущем шагу алгоритма

private List<int> mi;//Вектор количества партий данных для каждого типа данных

private List<int> np1i;//задание для каждого i-го типа данных количества решений по составам партий данных i-ых типов, сформированных на текущей ите-рации алгоритма

private List<int> np2i;//задание для каждого i-го типа данных количества решений по составам партий данных i-ых типов, сформированных на текущей ите-рации алгоритма

private List<List<int>> A1i;//Буферизированная матрица составов партий требований на k+1 шаге

private List<List<int>> A1;//Матрица составов партий требований на k+1 шаге

private List<List<int>> A2;//Матрица составов партий требований фиксированного типа

private List<List<int>> A;//Матрица составов партий требований на k шаге

private List<List<int>> ABuf;//Буферизированная матрица составов партий требований

private int countType;//количество типов

private List<int> countClaims;//Начальное количество требований для каждого типа данных

private BatchTypeClaims test;

private int i;//идентификатор текущего изменяемого типа

private int G;

private List<int> Gi;

private int q2;

private int q2i;

private int k;

private List<int> f1i;//Критерии начальных решений всех i-ого типа данных

private int f1;//Критерий текущего решения для всех типов

private int f1Buf;//Критерий текущего решения для всех типов

private bool solutionFlag;

/\*

\* Конструктор с параметрами

\*

\* count\_type - количество типов рассматриваемых данных

\* count\_claims - количество требований всех типов данных

\*

\*/

public FirstLevel(int count\_type, List<int> count\_claims)

{

this.countType = count\_type;

this.countClaims = count\_claims;

this.mi = new List<int>(this.countType);

this.np1i = new List<int>(this.countType);

this.np2i = new List<int>(this.countType);

this.I = new List<int>(this.countType);

this.Ii = new List<int>(this.countType);

this.Gi = new List<int>(this.countType);

}

/\*

\* Функция копирования значений между матрицами, предотвращающая копирование указателей

\*

\*/

private List<List<int>> CopyMatrix(List<List<int>> inMatrix)

{

List<List<int>> ret = new List<List<int>>();

for (int i = 0; i < inMatrix.Count; i++)

{

ret.Add(new List<int>());

for (int j = 0; j < inMatrix[i].Count; j++)

{

ret[i].Add(inMatrix[i][j]);

}

}

return ret;

}

/\*

\* Алгоритм формирования начальных решений по составам партий всех типов

\*

\*/

public void GenerateStartSolution()

{

int claim = 2;

this.A = new List<List<int>>();

this.A.Add(new List<int>());

for (int i = 1; i <= this.countType; i++)

{

this.I.Add(1);

this.Ii.Add(1);

this.Gi.Add(0);

this.mi.Add(claim);

this.np1i.Add(1);

this.np2i.Add(1);

this.A.Add(new List<int>());

this.A[i].Add(0);

this.A[i].Add(this.countClaims[i - 1] - claim);

this.A[i].Add(claim);

}

for (int i = 1; i <= this.countType; i++)

{

if (this.A[i][1] < 2 || this.A[i][1] < this.A[i][2])

{

this.A[i].Clear();

this.A[i].Add(0);

this.A[i].Add(this.countClaims[i - 1]);

this.mi[i - 1] = 1;

this.np1i[i - 1] = 0;

this.I[i - 1] = 0;

this.Ii[i - 1] = 0;

}

}

this.G = 0;

}

/\*

\* Функция вычисления f1 критерия

\*

\*/

public int GetCriterion(List<List<int>> inMatrix)

{

int criterion = 0;

for (int i = 1; i < inMatrix.Count; i++)

{

for (int j = 1; j < inMatrix[i].Count; j++)

{

criterion += inMatrix[i][j];

}

}

int criterionA = 0;

for (int i = 1; i < this.A.Count; i++)

{

for (int j = 1; j < this.A[i].Count; j++)

{

criterionA += this.A[i][j];

}

}

//return criterionA - criterion;

Random rand = new Random();

return rand.Next(7);

}

/\*

\* Функция проверок для реализации алгоритма

\*

\* flag == 1 - проверка на количество требований в соответствии с теоремой 1

\* flag == 2 - проверка на количество требований в первой партии в соответствии с теоремой 1

\*

\*/

public bool CheckingMatrix(int flag)

{

switch (flag)

{

case 1:

try

{

for (int i = 2; i < this.A2[1].Count; i++)

if (this.A2[1][1] < this.A2[1][i])

{

return false;

}

}

catch

{

return false;

}

break;

case 2:

try

{

if (this.A2[1][1] < 2)

{

return false;

}

}

catch

{

return false;

}

break;

}

return true;

}

/\*

\* Функция проверки наличия оставшихся в рассмотрении типов

\*

\*/

private bool CheckType(List<int> type)

{

int count = 0;

for (int j = 0; j < this.countType; j++)

{

if (type[j] != 0)

count++;

}

if (count == 0)

return true;

return false;

}

/\*

\* Алгоритм формирования решения по составам паритй всех типов данных

\*

\*/

public void GenerateSolution()

{

this.GenerateStartSolution();

SecondLevel secondLevel = new SecondLevel();

secondLevel.GenerateSolution(this.A);

List<List<int>> tmpMatrixA = secondLevel.ReturnAMatrix();

this.k = 0;

this.f1 = this.GetCriterion(tmpMatrixA);

this.f1Buf = this.f1;

//Добавить вычисление значения критерия

while (!this.CheckType(this.I))

{

this.solutionFlag = false;

//1 - Копируем I в Ii

for (int j = 0; j < this.countType; j++)

{

this.Ii[j] = this.I[j];

}

this.ABuf = this.CopyMatrix(A);

//Для каждого рассматриваемого типа

for (int iter = 0; iter < this.Ii.Count; iter++)

{

if (this.Ii[iter] != 0)

{

this.i = iter;

if (this.np1i[this.i] > 0)

{

this.A1 = this.CopyMatrix(this.A);

this.G = f1;

MessageBox.Show("Решение по составу партий данных " + (this.i + 1) + " типа на " + (this.k + 1) + " шаге алгоритма");

List<List<int>> toBatchAlgoritm = new   
 List<List<int>>();

toBatchAlgoritm.Add(new List<int>());

toBatchAlgoritm.Add(new List<int>());

toBatchAlgoritm[1] = this.A1[this.i + 1];

test = new BatchTypeClaims(this.f1,   
 this.i + 1, this.countClaims[this.i],   
 toBatchAlgoritm, this.A1);

test.GenerateSolution();

test.PrintMatrix(2);

test.PrintMatrix(3);

List<List<int>> tempA2 =   
 test.ReturnMatrix(3);

if (tempA2.Count < 2)

{

this.mi[this.i]++;

this.A2 = new List<List<int>>();

this.A2.Add(new List<int>());

this.A2.Add(new List<int>());

this.A2[1].Add(0);

this.A2[1].Add(0);

int sum = 0;

for (int j=1;j<this.mi[this.i]; j++)

{

this.A2[1].Add(2);

sum += 2;

}

this.A2[1][1] =   
 this.countClaims[this.i] - sum;

if (!this.CheckingMatrix(1) &&   
 !this.CheckingMatrix(2))

{

this.I[this.i] = 0;

continue;

}

}

else

{

this.A2 = this.CopyMatrix(tempA2);

}

this.A1i = this.CopyMatrix(this.A1);

for (this.q2 = 1; this.q2 <   
 this.A2.Count; this.q2++)

{

for (int h = 0; h <   
 this.A2[this.q2].Count; h++)

if (this.A1i[this.i + 1].Count <   
 this.A2[this.q2].Count)

{

this.A1i[this.i +   
 1].Add(this.A2[this.q2][h]);

}

else

{

this.A1i[this.i + 1][h] =   
 this.A2[this.q2][h];

}

secondLevel.GenerateSolution(this.A1i);

List<List<int>> tempMatrixA =   
 secondLevel.ReturnAMatrix();

int f1g =   
 this.GetCriterion(tempMatrixA);

if (f1g - this.f1Buf <= 0)

{

this.q2i = this.q2;

this.G = f1g - this.f1Buf;

}

}

if (this.G - this.f1 <= 0)

{

this.ABuf =   
 this.CopyMatrix(this.A1i);

this.f1Buf = this.G;

this.solutionFlag = true;

}

/\*if (count > 0)

{

this.q2i = 0;

for (int ind = 0; ind <   
 this.countType; ind++)

{

if (this.G > this.Gi[ind] &&   
 this.G < 0 && this.Gi[ind] < 0)

{

this.q2i = ind + 1;

this.G = this.Gi[ind];

}

}

if (this.q2i != 0)

{

this.A[this.i + 1] =   
 this.A2[this.q2i];

}

}\*/

}

}

}

this.k++;

if (this.solutionFlag)

{

this.f1 = this.f1Buf;

this.A = this.CopyMatrix(ABuf);

}

}

MessageBox.Show("Генерируем начальное решение");

List<List<int>> temp = new List<List<int>>();

temp.Add(new List<int>());

temp.Add(new List<int>());

temp[1].Add(0);

temp[1].Add(14);

temp[1].Add(2);

/\*temp.Add(new List<int>());

temp[2].Add(0);

temp[2].Add(5);

temp[2].Add(5);

temp[2].Add(3);

temp[2].Add(3);

temp.Add(new List<int>());

temp[3].Add(0);

temp[3].Add(5);

temp[3].Add(4);

temp[3].Add(4);

temp[3].Add(3);\*/

int countLoop = 0;

this.GenerateStartSolution();

return;

}

}

}

Код модуля построения групп партий фиксированного типа данных:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Diplom

{

/\*

\* BatchTypeClaims Class

\*

\* Необходим для реализации алгоритма получения решения в окресности О(k+1) из решений,

\* полученных в окресности О(k)

\*

\*/

class BatchTypeClaims

{

private List<List<int>> A;//Матрица составов партий требований максимальных решений

private List<List<int>> A1;//Матрица составов партий требований в окресности Ок

private List<List<int>> A2;//Матрица составов партий требований в окресности Ок+1

private int countClaims;//Начальное количество требований для текущего типа данных

private int np1;//Количество решений по составам партий данных , полученных на текущей (s+g)-ой итерации алгоритма

private int np2;//Количество решений по составам партий данных , полученных на последующей ((s+g)+1)-ой итерации алгоритма

private int q1;//Индекс решения по составам партий данных в А1

private int q2;//Индекс решения по составам партий данных в А2

private int q2i;//Индекс максимального решения по составам партий данных в А2

private int g;//Текущая итерация алгоритма

private int h;//Номер партии, состав которой будет изменяться

private int G;//Максимальное по модулю значение дискретного градиента

private int j;//Дополнительный индекс к номеру партии h

private int i;//Идентификатор текущего типа партии

private int f1;//критерий, полученный с первого уровня для определения лучшего решения

private bool inMatrixFlag = false;//флаг проверки входных зачений

/\*

\* Функция копирования значений между матрицами, предотвращающая копирование указателей

\*

\*/

private List<List<int>> CopyMatrix(List<List<int>> inMatrix)

{

List<List<int>> ret = new List<List<int>>();

for (int i = 0; i < inMatrix.Count; i++)

{

ret.Add(new List<int>());

for (int j = 0; j < inMatrix[i].Count; j++)

{

ret[i].Add(inMatrix[i][j]);

}

}

return ret;

}

/\*

\* Функция печати результатов алгоритма

\*

\*/

public void PrintMatrix(int a = 1)

{

string s = "";

List<List<int>> matrix = new List<List<int>>();

switch (a)

{

case 1:

s += "Матрица А\n";

matrix = this.A;

break;

case 2:

s += "Матрица А1\n";

matrix = this.A1;

break;

case 3:

s += "Матрица А2\n";

matrix = this.A2;

break;

}

foreach (List<int> row in matrix)

{

foreach (int colum in row)

{

s += colum + ", ";

}

s += "\n";

}

MessageBox.Show(s);

//MessageBox.Show(s);

}

/\*

\* Возврат полученные решения по составам партий из различных матриц

\*

\* param == 1 - Возврат матрицы А

\* param == 2 - Возврат матрицы А1

\* param == 3 - Возврат матрицы А2

\*

\*/

public List<List<int>> ReturnMatrix(int param)

{

List<List<int>> matrix = new List<List<int>>();

switch (param)

{

case 1:

matrix = this.A;

break;

case 2:

matrix = this.A1;

break;

case 3:

matrix = this.A2;

break;

}

List<List<int>> ret = new List<List<int>>();

for (int i = 0; i < matrix.Count; i++)

{

ret.Add(new List<int>());

for (int j = 0; j < matrix[i].Count; j++)

{

ret[i].Add(matrix[i][j]);

}

}

return ret;

}

/\*

\* Конструктор с параметрами

\*

\* valueI - идентификатор текущего типа, подлежащего изменению

\* valueCountClaims - количество требований текущего типа

\* valueA1 - начальные решения, полученные на предыдущем шаге (текущее локальное решение)

\* valueA - матрица решений по всем типам данных (текущее глобальное решение)

\*

\*/

public BatchTypeClaims(int valieCriterion, int valueI, int valueCountClaims, List<List<int>> valueA1, List<List<int>> valueA)

{

/\*this.A1 = new List<List<int>>();

this.A1.Add(new List<int>());

this.A1.Add(new List<int>());

this.A1[1] = valueA[valueI];\*/

this.A = this.CopyMatrix(valueA);

this.A1 = this.CopyMatrix(valueA1);

this.A2 = new List<List<int>>();

try

{

this.A2.Add(new List<int>());

this.A2.Add(new List<int>());

for (int i = 0; i < valueA1[1].Count; i++)

{

this.A2[1].Add(0);

}

}

catch

{

this.inMatrixFlag = true;

}

this.i = valueI;

this.np2 = 0;

this.np1 = this.A1.Count - 1;

this.q2 = 0;

this.q2i = 0;

this.g = 1;

this.f1 = valieCriterion;

this.countClaims = valueCountClaims;

}

/\*

\* Функция проверок для реализации алгоритма

\*

\* flag == 1 - проверка на количество требований в первой партии в соответствии с теоремой 1 на шаге 4

\* flag == 2 - проверка на количество требований в соответствии с теоремой 2 на шаге 5

\* flag == 3 - проверка на количество требований в соответствии с теоремой 2 на шаге 6

\*

\*/

public bool CheckingMatrix(int flag)

{

switch (flag)

{

case 1:

try

{

for (int i = 2; i < this.A2[this.q2].Count; i++)

if (this.A2[this.q2][1] <   
 this.A2[this.q2][i])

{

return false;

}

}

catch

{

return false;

}

break;

case 2:

try

{

if (this.A1[this.q1][this.h] >   
 this.A1[this.q1][this.h + this.j])

{

return true;

}

}

catch

{

return false;

}

break;

case 3:

try

{

if (this.A1[this.q1][this.h] ==   
 this.A1[this.q1][this.h + this.j])

{

return true;

}

}

catch

{

return false;

}

break;

}

return true;

}

/\*

\* Функция получения неповторяющихся решений в матрице А2 на шаге 9

\*

\*/

public List<List<int>> SortedMatrix(List<List<int>> inMatrix)

{

List<List<int>> temp = this.CopyMatrix(inMatrix);

//сортировка буферной матрицы

foreach (List<int> row in temp)

{

row.Sort(delegate(int mc1, int mc2)

{

return mc1.CompareTo(mc2);

});

}

//Удаление повторяющихся строк

int countLoops = 0;

while (true)

{

for (int i = 1; i < temp.Count; i++)

{

int lastIndexForDelete =   
 temp.FindLastIndex(delegate(List<int> inList)

{

int count\_find = 0;

for (int k = 0; k < inList.Count; k++)

{

if (inList[k] == temp[i][k])

{

count\_find++;

}

}

return count\_find == inList.Count ? true : false;

});

if (lastIndexForDelete != i)

{

temp.RemoveAt(lastIndexForDelete);

inMatrix.RemoveAt(lastIndexForDelete);

}

}

countLoops++;

if (countLoops > 100)

break;

}

return inMatrix;

}

/\*

\* Функция вычисления f1 критерия

\*

\*/

public int GetCriterion(List<List<int>> inMatrix)

{

int criterion = 0;

for (int i = 1; i < inMatrix.Count; i++)

{

for (int j = 1; j < inMatrix[i].Count; i++)

{

criterion += inMatrix[i][j];

}

}

int criterionA = 0;

for (int i = 1; i < this.A.Count; i++)

{

for (int j = 1; j < this.A[i].Count; j++)

{

criterionA += this.A[i][j];

}

}

return criterionA - criterion;

}

/\*

\* Формирование нового решения на основе рассматриваемого

\*

\*Используются индексы q1 и q2

\*q1 - индекс текущего решения в матрице А1

\*q2 - индекс текущего решения в матрица А2

\*В результате происходит заполнение матрицы А2 значениями, полученными на основе матрицы А1

\*

\*/

public void FormationDecisionPartMakeup()

{

if (this.q2 >= this.A2.Count)

{

this.A2.Add(new List<int>());

for (int i = 0; i < this.A1[this.q1].Count; i++)

{

this.A2[this.q2].Add(0);

}

}

int summ = 0;

if (this.q1 < this.A1.Count)

{

for (int i = 1; i < this.A1[this.q1].Count; i++)

{

if (i != this.h)

{

this.A2[this.q2][i] = this.A1[this.q1][i];

}

else

{

this.A2[this.q2][i] = this.A1[this.q1][i] + 1;

}

if (i != 1)

summ += this.A2[this.q2][i];

}

this.A2[this.q2][1] = this.countClaims - summ;

}

}

/\*

\* Основная функция работы алгоритма

\*

\*/

public bool GenerateSolution()

{

if (this.inMatrixFlag)

{

return false;

}

bool step5 = false;

this.q1 = 1;

this.h = 2;

this.j = 1;

while (this.q1 <= this.np1)

{

if (!step5)

{

step5 = false;

this.q2++;

this.FormationDecisionPartMakeup();

this.np2++;

if (!this.CheckingMatrix(1))

{

this.A2.RemoveAt(this.q2);

this.q2--;

this.np2--;

}

}

if (this.h + this.j <= this.countClaims)

{

if (this.CheckingMatrix(2))

{

this.h += this.j;

continue;

}

if (this.CheckingMatrix(3))

{

this.j++;

step5 = true;

continue;

}

}

this.q1++;

this.h = 2;

this.j = 1;

}

if (this.np2 > 1)

{

this.A2 = this.SortedMatrix(this.A2);

this.np2 = this.A2.Count - 1;

this.q2i = 0;

this.G = 0;

if (this.np2 > 0)

{

for (int indexQ = 1; indexQ < this.A2.Count;   
 indexQ++)

{

List<List<int>> tempA = CopyMatrix(this.A);

tempA[this.i] = this.A2[indexQ];

SecondLevel secondLevel = new SecondLevel();

if (secondLevel.GenerateSolution(tempA))

{

List<List<int>> tempMatrixA =   
 secondLevel.ReturnAMatrix();

int f1g =   
 this.GetCriterion(tempMatrixA);

if (f1g - this.f1 <= 0)

{

if (f1g - this.f1 < this.G)

{

this.q2i = indexQ;

this.G = f1g - this.f1;

}

}

}

}

}

else

{

return false;

}

}

return true;

}

}

}

# ПРИЛОЖЕНИЕ Б

**Расчеты и таблицы экономического раздела**

Этапы маркетинговых исследований:

Процесс маркетинговых исследований состоит из шести этапов:

1. Определение проблемы. Первый этап любого маркетингового исследования заключается в выяснении проблемы. При ее определении маркетолог должен учитывать цель исследования, соответствующую исходную информацию, которая необходима, и как она будет использована при принятии решения.

2. Разработка плана по решению проблемы. Разработка плана по решению проблемы включает формулировку теоретических рамок исследования, аналитических моделей, поисковых вопросов, гипотез, а также определение факторов, которые могут влиять на план исследования.

3. Разработка плана исследования. План маркетингового исследования детализирует ход выполнения процедур, необходимых для получения нужной информации. Он необходим для того, чтобы разработать план проверки гипотез, выяснить, какая информация необходима для принятия решения.

4. Полевые работы или сбор данных. Разработав программу исследования, необходимо собрать информацию. Как правило, это самый дорогой и самый чреватый ошибками этап исследования. В зависимости от объема выборки требуется определенное количество обученных работников, владеющих технологией сбора информации.

5. Подготовка данных и их анализ. Следующий этап маркетингового исследования – извлечение из совокупности полученных данных наиболее важных сведений и результатов. Сложность анализа качественной информации в том, что результаты могут интерпретироваться по-разному, в зависимости от квалификации аналитика.

6. Подготовка отчета и его презентация. Ход и результаты маркетинговых исследований должны быть изложены письменно в виде отчета, в котором четко обозначены конкретные вопросы исследования, описанный метод и план исследования, процедуры сбора данных и их анализа, результаты и выводы.

Таблица Б.1 – Оценка шансов и рисков проектируемого продукта

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Показатели | Балы | | | | | | | | |
| Опасность | | | Нейтрально | | | Шансы | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 1. Объем рынка |  |  |  |  |  |  | + |  |  |
| 2. Рост рынка |  |  |  |  |  |  | + |  |  |
| 3. Финансовый потенциал |  |  |  |  | + |  |  |  |  |
| 4. Число конкурентов |  |  |  |  |  | + |  |  |  |
| 5. Поведение конкурентов |  |  |  |  |  | + |  |  |  |
| 6. Осведомленность потребителя |  |  |  |  |  | + |  |  |  |
| 7. Возможность повышения цен |  |  |  | + |  |  |  |  |  |
| 8. Изменение конъюнктуры рынка |  |  |  |  |  | + |  |  |  |
| 9. Возможность замещения продукта |  |  |  |  |  |  | + |  |  |
| 10. Потенциал сервиса |  |  |  |  |  |  |  | + |  |

Таблица Б.2 – Список потенциальных организаций и предприятий Крыма

| Название | Область деятельности |
| --- | --- |
| [ООО «Гермес](http://alliance-catalog.ru/da_trans/)» | ООО «Гермес», поставляет оборудование для предприятий конвейерного типа |
| ООО «СИР» | ООО «СИР» предлагает к поставке конвейерную продукцию для использования в различных областях промышленности Крымского региона |
| [«Евпаторийский Винный Завод](http://tavrika.su/id15175)» | Производство вина и его розлив с помощью оборудования конвейерного типа. |
| [Картонно-тарный комбинат](http://tavrika.su/id24838) | Изготовление тары из гофрокартона,производство конвейерного типа. |
| [Крымский кирпичный завод](http://tavrika.su/id27748) | Призводство кирпичей. |
| [ХимRussia](http://tavrika.su/id28574) | Производство автохимии и средств для профессионального клининга в Крыму. |

Продолжение таблицы Б.2

| Название | Область деятельности |
| --- | --- |
| Компания "Эко безопасность Крым" | Производство тентовых накрытий и каркасов для летних площадок, кафе, ресторанов, дискотек и концертных площадок - тенты и каркасы для грузовых автомобилей. |
| Производство "Абсолют-мебель" | Мебельная фабрика "Абсолют-мебель" специализируется на производстве детских парт, детских комнат, а также корпусной мебели для дома, офиса |
| ООО ПКФ "Черемош" | Изготовление фурнитуры для мебели |
| Фирма "Планета Цвета" | Мебельные фасады из МДФ с любым изображением; Стекло ламинированное пластиком; Стекла с тканью или обоями внутри. |

Таблица Б.3 – Функции ПП

| Наименование функции | Объем функций |
| --- | --- |
| организация ввода информации | 0,335 |
| контроль | 1,05 |
| управление вводом/выводом | 3,225 |
| формирование служебных таблиц | 2,005 |
| монитор ПС ВТ (управление работой компонентов) | 1,555 |
| обработка входного заказа и формирование таблиц | 0,775 |
| обеспечение интерфейса между компонентами | 3,43 |
| обработка ошибочных и сбойных ситуаций | 2,6 |
| справка и обучение | 0,225 |

Таблица Б.4 – Расчет показателей трудоемкости

| Показатель трудоемкости | Формула | Коэффициент | Полученное значение |
| --- | --- | --- | --- |
| ТЗ | *Lтз*×*Kн*×*Tо*=0,11×1,0×347,26 | 0,11 | 31,14 |
| ЭП | *Lэп*×*Kн*×*Tо*=0,09×1,0×347,26 | 0,09 | 25,48 |
| ТП | *Lтп*×*Kн*×*Tо*=0,11×1,0×347,26 | 0,11 | 31,14 |
| РП | *Lрп*×*Kн*×*Tо*×*Kт*=0,55×1,0×347,26×1,0 | 0,55 | 109,00 |
| ВН | *Lвн*×*Kн*×*Tо*=0,14×1,0×347,26 | 0,14 | 39,64 |
| Всего | ΣLi | 1,00 | 236,41 |

Таблица Б.5 – Данные для расчета годовых эксплуатационных затрат

| Основные показатели | Условное обозначение | Единица измерения | Значение показателя |
| --- | --- | --- | --- |
| Стоимость основного комплекта ЭВМ | *С* | руб. | 21000,00 |
| Потребляемая мощность | *W* | кВТ/ч | 0,35 |
| Коэффициент использования по мощности | *Kи* |  | 0,70 |
| Цена 1 кВт/ч электроэнергии | *Цэ* | руб. | 2,53 |
| Номинальный фонд времени работы ЭВМ | *Fном* | час | 1974,00 |
| Потери времени на ремонт и профилактику (% от Fном) | *Ппот* | % | 4,00 |
| Коэффициент годовых затрат на ремонт (от стоимости ЭВМ) | *Кр* | % | 6,00 |
| Коэффициент сменности | *Ксм* |  | 1,00 |
| Норма амортизационных отчислений на ЭВМ | *Ноб* | % | 4,17 |
| Коэффициент начислений на фонд оплаты труда | *Кн* | % | 34,20 |
| Коэффициент накладных расходов (% от ФОТ) | *Кнр* | % | 22,00 |
| Коэффициент материальных затрат (% от стоимости ЭВМ) | *Кмз* | % | 7,00 |
| Оклад разработчика | *Окр* | Руб. | 9500,00 |

Таблица Б.6 – Смета годовых эксплуатационных затрат

| Наименование затрат | Формула расчета | Значение, руб |
| --- | --- | --- |
| Расчет материальных затрат | *Зм = С × Кмз* / 100 = 21000 × 0,07 | 1470 |
| Расчет затрат на электроэнергию | *Зэ* *= Fном × Цэ × W × Кн × Ксм* = 1974 × 2,53 × 0,35 × 1 | 1223,5839 |
| Расчет оплаты труда | *ФОТ* = *Срр × Ксм × Окр* = 11,8 × 1 × 9500,00 | 112100,00 |
| Расчет отчислений от заработной платы | *Отч* = *ФОТ × Кн* / 100 = 112100 × 0,342 | 38338,20 |
| Расчет затрат на ремонт | *Зр* = *С × Кр* / 100 = 21000 × 0,06 | 1260,00 |
| Расчет накладних расходов | *Зн* = (*Зм + Зэ + ФОТ + Отч + Зр*) × *Кнр* / 100 = (1470 + 1223,5839 + 112100 + 38338,2 + 1260) × 0,22 | 33966,19 |
| Расчет амортизационных отчислений | *Аоб* = *С×Ноб* × 11,8/100 = 21000 × 4,17 × 11,8/100 | 10325,00 |
| Всего |  | 198682,98 |

Таблица Б.7 – Смета затрат на проектирование

| Наименование расходов | Формула расчетов | Значение, руб |
| --- | --- | --- |
| Основная заработная плата разработчика | *Зп.осн*=*Ч x Срр x Оклад* = 1 × 0,44 × 9500 | 4154,53 |
| Фонд оплаты труда проектирования (за работы, выполняемые без ЭВМ) | *ФОТ* = *Зп.осн + Зп.осн × Кдоп.з/п* = 4154,53 + 4154,53 × 0,3 | 5400,89 |
| Фонд оплаты труда с отчислениями | *ФОТотч* = *ФОТ + Отчисление* = 5400,89 + 5400,89 × 0,342 | 7247,99 |
| Накладные расходы | *Знакл* = *Кнр × ФОТотчисл*/100 = 7247,99 × 0,22 | 1594,56 |
| Затраты на разработку ПП с применением ВТ | *Зпо* = *Срм × Fэф × Сч.м.в* = 1 × 168 × 104,3 | 17522,48 |
| Затраты на материалы | *Зм* | 1500,00 |
| Всего |  | 27865,03 |

Таблица Б.8 – Формирование цены предложения разработчика

| Наименование расходов | Формула расчетов | Значение, руб |
| --- | --- | --- |
| Капитальные затраты | *Кр* | 26230,00 |
| Постоянные издержки | *Ипост*=*Σзатр проектир* | 27865,03 |
| Затраты на послепродажное обслуживание | *Зо* | 6600,00 |
| Затраты на рабочую документацию | *Зрд* | 3000,00 |
| Затраты на защиту | *Ззащ* | 7000,00 |
| Затраты на рекламу | *Зрек*= 25% × *Зпроек* = 0,25 × 27865,03 | 6966,26 |
| Переменные издержки | *Ипер*=*Зрек+Зрд+Ззащ+Зо*= 6966,26+ 3000 + 7000 + 6600 | 23566,26 |
| Полные издержки | *Иполн*=*Ипост+Ипер* = 27865,03 + 23566,26 | 51431,28 |
| Прибыль | *Прибыль*=*Иполн* × 40% = 0,4 × 51431,28 | 20572,51 |
| Выручка | *В*=*Иполн+Прибыль* = 51431,28 + 20572,51 | 72003,80 |
| Количество компаний | N | 10,00 |
| Цена ед. товара | *Ц*ед=*B/N* = 72003,80 / 10,00 | 7200,38 |

Таблица Б.9 – Расчет интегрального экономического эффекта

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Наименование расчета | Формула | Годы | | | | |
| Год |  | 2016,00 | 2017,00 | 2018,00 | 2019,00 | 2020,00 |
| Проданное количество | *Nt* | 0,00 | 10,00 | 9,00 | 8,00 | 6,00 |
| Выручка | *Pt=Цед × Nt* | 0,00 | 72003,80 | 64803,42 | 57603,04 | 43202,28 |
| Капитальные вложения | *Kt* | 26230,00 | 0,00 | 0,00 | 0,00 | 0,00 |
| Издержки | *Иt=Иполн × kt (годовой коэффициент)* | 0,00 | 51431,28 | 41145,03 | 36001,90 | 25715,64 |
| Чистый денежный поток | *ЧДП=Pt‑*(*Kt+Иt*) | -26230,00 | 20572,51 | 23658,39 | 21601,14 | 17486,64 |
| Показатель текущей стоимости | *αt=1 /* (*1+E*)^*t* | 1,00 | 0,80 | 0,71 | 0,64 | 0,57 |
| Чистая текущая стоимость | *ЧТС=ЧДП × α* | -26230,00 | 16400,28 | 16839,57 | 13727,91 | 9922,39 |
| Интегральный экономический эффект | *ИЭЭt=ИЭЭt-1 + ЧТСt* | -26230,00 | -9829,72 | 7009,86 | 20737,77 | 30660,16 |
| Амортизация | *At=Кр* */ 5* | 5246,00 | 5246,00 | 5246,00 | 5246,00 | 5246,00 |
| Прибыль | *Прt=Pt‑Иt‑At* */ 4* | 0,00 | 15326,51 | 18412,39 | 16355,14 | 12240,64 |
| Рентабельность | *Рo=Прt* */ Иt* | 0,00 | 29,80 | 44,75 | 45,43 | 47,60 |



Рисунок Б.1 – Жизненный цикл ПП



Рисунок Б.2 – Трехуровневая модель товара

# ПРИЛОЖЕНИЕ В

**Расчет показателей экономической эффективности**

Порядок расчета показателей экономической эффективности следующий:

1. Определение показателей чистого денежного потока (*ЧДП*) за период реализации проекта по формуле:

,

где - чистый денежный поток года t, руб.;

– объем реализации выручка от реализации работ и услуг в году t, руб

– капитальные вложения года t, руб.;

– издержки года t (без амортизационных отчислений) в году, руб.

Объем реализации работ для разработчика определяется следующим образом:

где ** – годовой объем реализации изделий (пакетов программ), шт;

– цена реализации одного изделия (пакета программ) в году, руб.

Годовые издержки:

Чистый денежный поток:

2. Определение показателей чистой текущей стоимости за период реализации проекта по формуле:

где *E*– ставка дисконтирования или норма доходности (прибыльности) от вложения средств(будем считать *Е* = 0,12); – расчетный год;

Рассчитаем показатели дисконтированного чистого денежного потока (*ДЧДП*)

3. Определение интегрального экономического эффекта

4. Рассчитаем амортизацию по формуле

где капитальные вложения

*T* – года; *T* = 5 (лет.)

Амортизация по годам распределяется прямолинейным методом.

5. Определим показатели прибыли

где, - объем реализации,

- годовые издержки

6. Найдем рентабельность по формуле:

# ПРИЛОЖЕНИЕ Г

**Схемы помещений и показатели охраны труда**

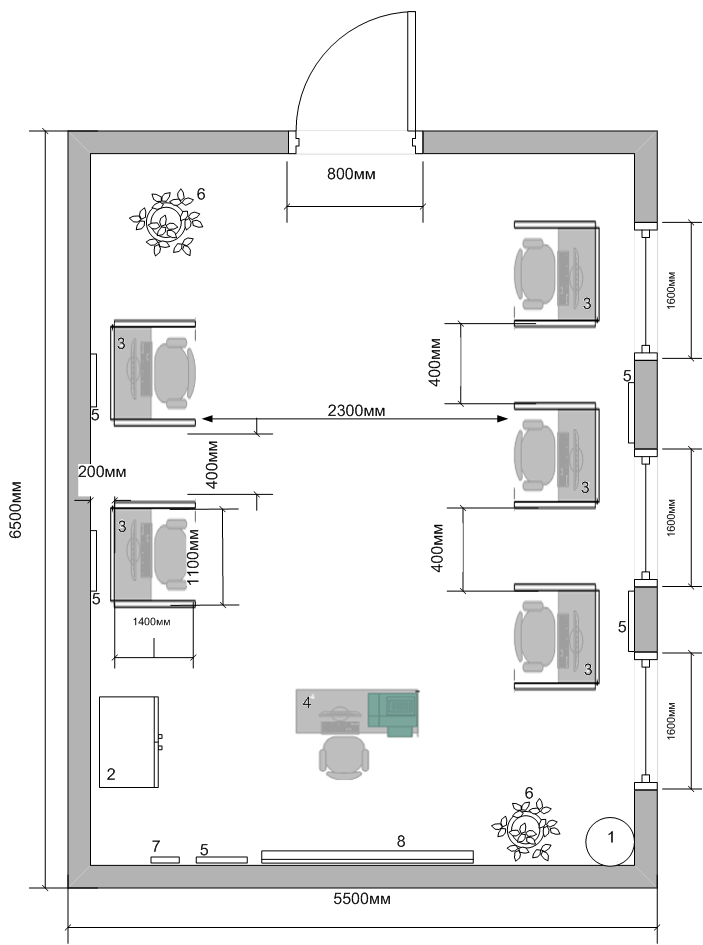


Рисунок Г.1 – Схема помещения

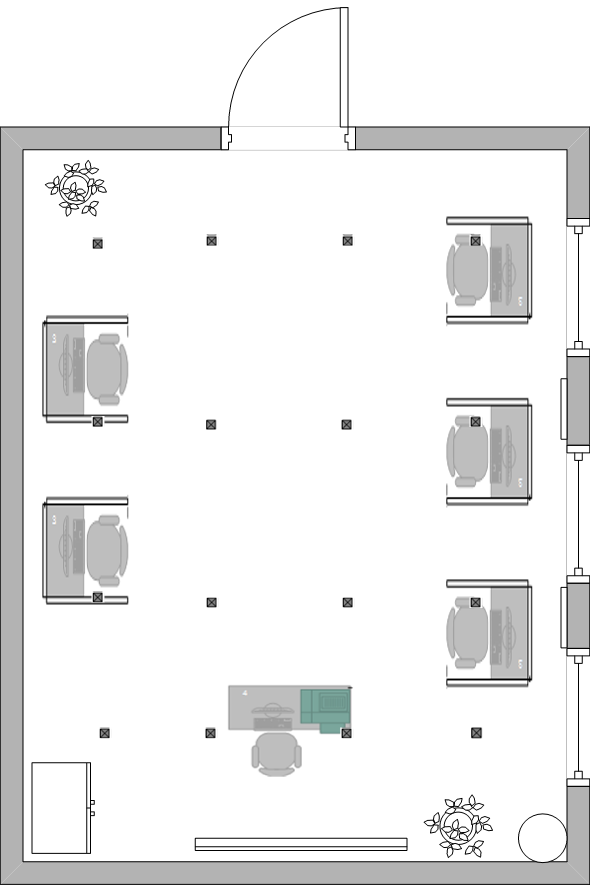


Рисунок Г.2 – Схема размещения искусственного освещения

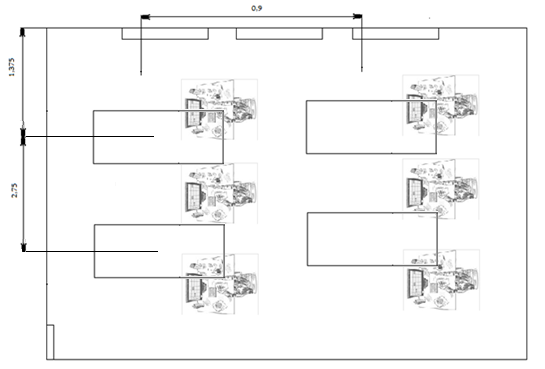


Рисунок Г.3 –Схема расположения рабочих мест



1- стол; 2-стул; 3-подставка для ног

Рисунок Г.4 – Схема рабочего места.

Таблица Г.1 – Карта условий труда

| Фактор производственной среды или трудового процесса | Дата исследования | Нормативное значение (ПДК, ПДУ, допустимое значение) | Фактическое (измеренное) значение | Продолжительность действия фактора (часов или %) | Класс условий труда |
| --- | --- | --- | --- | --- | --- |
| Тяжесть трудового процесса | | | | | |
| Рабочая поза |  | Свободная удобная поза, возможность изменения (сидя, стоя) по желанию работника | Периодические пребывания в неудобной позе (работа с поворотом туловища, неудобным расположением конечностей) | до 25% времени смены | 2 |
| Напряженность трудового процесса | | | | | |
| Интеллектуальные нагрузки | | | | | |
| Содержание работы |  | Отсутствие необходимости принимать решения | Решение сложных задач с выбором по известным алгоритмам (работа по серии инструкций) | до 90% времени смены | 3.1 |
| Характер выполняемой работы |  | Работа по индивидуальному плану | Работа по установленному графику с возможным его корректированием в ходе деятельности |  | 2 |
| Сенсорные нагрузки | | | | | |
| Продолжительность сосредоточенного наблюдения (в % от времени смены) |  | > 75 % | 70% |  | 2 |
| Плотность сигналов (световых, звуковых) и сообщений в среднем за 1 час работы |  | 75-175 | 80 |  | 2 |
| Количество производственных объектов одновременного наблюдения |  | до 5 | 1-2 |  | 1 |
| Нагрузка на зрительный анализатор |  | более 5мм 100% времени | 1,0-0,3 мм  больше 50% времени |  | 3.1 |

Продолжение таблицы Г.1

| Фактор производственной среды или трудового процесса | Дата исследования | Нормативное значение (ПДК, ПДУ, допустимое значение) | Фактическое (измеренное) значение | Продолжительность действия фактора (часов или %) | Класс условий труда |
| --- | --- | --- | --- | --- | --- |
| Размер объекта различения (при расстоянии от глаз работающего до объекта различения не более 0,5 м) в мм при длительности сосредоточенного наблюдения (% времени смены) |  | более 5 мм | 1.0-0.3 мм до 50%времени меньше 0.3 мм до 25% | 0.2 | 2 |
| Наблюдение за экранами видеотерминалов (часов в смену) |  | до 2 | 2-3 | 3 | 2 |
| Эмоциональные загрузки | | | | | |
| Степень ответственности за результат собственной деятельности. Значимость ошибок |  | Несет ответственность за выполнение отдельных элементов задания | Несет ответственность за выполнение отдельных элементов задачи. Ошибка требует дополнительных усилий работника. |  | 1 |
| Монотонность загрузок | | | | | |
| Количество элементов (приемов), необходимых для реализации простой задачи или в операциях повторяющихся многократно |  | более 10 | больше 10 | 12 | 1 |
| Продолжительность выполнения простых производственных задач или операций, которые повторяются, сек |  | более 100 с | Более 100 с | 120с | 1 |
| Время активных действий (в % к продолжительности смены) Остаток времени - наблюдение за технологическим процессом |  | 20 и более | 20 и более | 24 | 1 |

Продолжение таблицы Г.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Фактор производственной среды или трудового процесса | Дата исследования | Нормативное значение (ПДК, ПДУ, допустимое значение) | Фактическое (измеренное) значение | Продолжительность действия фактора (часов или %) | Класс условий труда |
| Монотонность производственной обстановки (время пассивного наблюдения за технологическим процессом в % от времени смены) |  | меньше 75 | меньше 75 | 50 | 1 |
| Режим работы | | | | | |
| Фактическая продолжительность рабочего дня (ч.) |  | 8-9 часов | 8-9 часов | 8 часов | 2 |
| Сменность работы |  | Односменная работа (без ночной смены) | Односменная работа (без ночной смены) |  | 1 |
| Наличие регламентированных перерывов и их продолжительность |  | Перерывы регламентированные, достаточной продолжительности 7% и больше времени смены | Перерывы регламентированные, достаточной продолжительности 7% и больше времени смены | 10 минут каждый час | 1 |

Таблица Г.2 – Нормы микроклимата для помещений с ПЭВМ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Период года | Категории работ | Температура воздуха, ˚С | Относительная влажность воздуха, % | Скорость движения воздуха, м/с |
| Холодный | Легкая 1а | 22-24 | 40-60 | 0,1 |
| Легкая 1б | 21-23 | 40-60 | 0,1 |
| Теплый | Легкая 1а | 23-25 | 40-60 | 0,1 |
| Легкая 1б | 22-24 | 40-60 | 0,2 |

Таблица Г.3 – Состав воздуха на рабочих местах

|  |  |  |
| --- | --- | --- |
| Вещество | ПДК | Измеренное значение |
| Оксиды азота | 5 мг/м3 | 3,4 мг/м3 |
| Пыль | 4 мг/м3 | 3,6 мг/м3 |
| Озон | 0,1 мг/м3 | 0,05 мг/м3 |

Таблица Г.4 - Допустимые уровни звукового давления и уровня звука

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Рабочие места | допустимые уровни звукового давления,дБ,на рабочем месте в октавных полосах со средними геометрическими,Гц | | | | | | | | | Уровни звука, дБА |
|  | 31,5 | 63 | 125 | 250 | 500 | 1000 | 2000 | 4000 | 8000 |  |
| Рабочее место программиста | 86 | 71 | 61 | 54 | 49 | 45 | 42 | 40 | 38 | 35 |

Таблица Г.5- Нормы по освещению рабочего помещения

|  |  |  |
| --- | --- | --- |
| Тип помещения | Средняя освещенность по нормам (лк) | Нормы коэффициента естественной освещенности (КЕО) в помещениях |
| Помещение общего назначения с использованием компьютеров | 300 -500 | 1,2%-1,6% |
| Фактическое значение конкретного помещения | 350 | 1,4% |

Таблица Г.6 – Классы пожаров

|  |  |  |  |
| --- | --- | --- | --- |
| Класс пожара | Характеристика класса | Подкласс пожара | Характеристика класса |
| А | Горение твердых веществ | А1 | Горение твердых веществ, сопровождаемое тлением (древесина, бумага) |
| E | Горение объектов, находящихся под напряжением | - | Горение установок и оборудования, находящихся под электрическим напряжением |

Таблица Г.7 - Параметры рабочего места пользователя ПЭВМ

| Элемент рабочего места | Параметры | Обозначение по рис.2.2 | Фактическая величина | Нормативное значение |
| --- | --- | --- | --- | --- |
| 1.Рабочий стол | Рабочая поверхность:  -высота | Н | 800 | 680-800мм |
| -ширина | - | 1200 | 800-1400мм |
| -глубина | В | 800 | 800-1400мм |
| Пространство для ног  -высота | H | 650 | Не менее: 600мм |
| -ширина | - | 700 | 500мм |
| -глубина на уровне колен | - | 500 | 450мм |
| -глубина на уровне вытянутых ног | a2 | 650 | 650мм |

Продолжение таблицы Г.7

| Элемент рабочего места | Параметры | Обозначение по рис.2.2 | Фактическая величина | Нормативное значение |
| --- | --- | --- | --- | --- |
| 2.Рабочий стул  (подъемно-поворотный) | Ширина сиденья | b | 500 | =>400мм |
| Глубина сиденья | a | 500 | =>400мм |
| Высота поверхности сиденья | h1 | 400 | 400-550мм |
| Угол наклона сиденья  -вперед  -назад | -  - | 100  50 | Не более: 150 50 |
| Высота опорной поверхности спинки | - | 300 | =>300мм |
| Радиус кривизны спинки в горизонтальной  плоскости | с | 400 | 260-400мм |
| Подлокотник (съемные или стационарные): |  |  |  |
| -длина | L | 300 | =>250мм |
| -ширина | b2 | 50 | 50-70мм |
| -высота на сиденье | h3 | 250 | 200-260мм |
| -расстояние между подлокотниками | T | 400 | 350-500мм |
| 3.Подставка для ног | Ширина | bп | 350 | =>300мм |
| Глубина | aп | 500 | =>400мм |
| Высота | hп | 100 | <=150мм |
| Наклон опорной поверхности | - | 15 | <=200 |
| 4.Пюпитр для документов  (перемещаемый) | Наличие | - |  | Имеется |