МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)
Кафедра информационных систем и технологий

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«__» _____ 2017г.

Проектирование программных систем на основе современных технологий программирования

Методические указания к лабораторному практикуму по дисциплине «Технологии программирования» для студентов, обучающихся по направлениям 09.03.02 «Информационные системы и технологии» и 09.03.03 «Прикладная информатика»

Курск 2017

УДК 004
Составитель И.В. Зотов

Рецензент
Кандидат технических наук Ю.А. Халин

Проектирование программных систем на основе современных технологий программирования: методические указания к лабораторному практикуму / Юго-Зап. гос. ун-т; сост. И.В. Зотов. Курск, 2017. 60 с. Библиогр.: с. 60.

Рассматривается методика проектирования программных систем средней сложности на основе использования метода пошаговой детализации в рамках объектной технологии программирования. Изложение сопровождается примером проектирования программы имитационного моделирования системы массового обслуживания. Приводятся варианты заданий к лабораторному практикуму.

Методические указания предназначены для студентов, обучающихся по направлениям 09.03.02 «Информационные системы и технологии» и 09.03.03 «Прикладная информатика». Могут использоваться также студентами, обучающимися по направлениям, связанным с вычислительной техникой и интеллектуальными информационными системами.

Текст печатается в авторской редакции.

Подписано в печать . Формат 60x84 1/16.
Усл.печ. л. . Уч.-изд. л. . Тираж ___ экз. Заказ. Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

1. Цели лабораторного практикума

Целями лабораторного практикума являются:

- освоение приемов проектирования программных систем средней сложности на основе использования метода пошаговой детализации;
- овладение навыками работы в современных средах быстрой разработки программных систем с применением визуального программирования и компонентного подхода;
- приобретение умений в использовании современных технологий отладки и тестирования программ, стратегий интеграции программных систем.

2. Краткие теоретические сведения

Технологией программирования называют совокупность методов и средств, используемых в процессе разработки программного обеспечения. Как любая другая технология, технология программирования представляет собой набор технологических инструкций, включающих:

- указание последовательности выполнения технологических операций;
- перечисление условий, при которых выполняется та или иная операция;
- описания самих операций, где для каждой операции определены исходные данные, результаты, а также инструкции, нормативы, стандарты, критерии и методы оценки и т. п.

Кроме набора операций и их последовательности, технология также определяет способ описания проектируемой системы, точнее модели, используемой на конкретном этапе разработки.

Различают технологии, используемые на конкретных этапах разработки или для решения отдельных задач этих этапов, и технологии, охватывающие несколько этапов или весь процесс разработки. В основе первых, как правило, лежит ограниченно применимый *метод*, позволяющий решить конкретную задачу. В основе вторых обычно лежит базовый метод или *подход*, определяющий совокупность методов, используемых на разных этапах разработки, или *методологию*.

Большинство современных программных систем (ПС) объективно очень сложны и их разработка без использования современных технологий программирования невозможна. Сложность ПС обусловлена многими причинами, главной из которых является *логическая сложность решаемых* ими *задач.* Также существует ряд факторов, дополнительно увеличивают сложность современных ПС. К их числу относятся:

- сложность формального определения требований к ПС;
- отсутствие адекватных средств описания поведения дискретных систем с большим числом состояний при недетерминированной последовательности входных воздействий;
- коллективная разработка;
- необходимость увеличения степени повторяемости кодов.

Практика показывает, что подавляющее большинство сложных систем, как в природе, так и в технике имеет иерархическую внутреннюю структуру. Это связано с тем, что обычно связи элементов сложных систем различны как по типу, так и по силе, что и позволяет рассматривать эти системы как некоторую *совокупность взаимозависимых подсистем*. Внутренние связи элементов таких подсистем сильнее, чем связи между подсистемами. Сказанное в полной мере относится и к программным системам.

Иерархическая природа сложных ПС позволяет использовать при их создании *блочно-иерархический подход*. Этот подход предполагает, что сначала изготавливаются части ПС (блоки, модули), а затем из них собирается сама система, т.е. предполагается разбиение системы на составляющие.

Процесс разбиения сложной системы на сравнительно независимые части получил название *декомпозиции*. При декомпозиции необходимо стремиться, чтобы связи между отдельными частями должны быть слабее, чем связи элементов внутри частей. Кроме того, чтобы из полученных частей можно было собрать разрабатываемую систему, в процессе декомпозиции необходимо определить все виды связей частей между собой.

При создании очень сложных ПС процесс декомпозиции выполняется многократно: каждый блок, в свою очередь, декомпозируют на части, пока не получат блоки, которые сравнительно легко разработать. Данный метод разработки получил название *пошаговой детализации.*

Результат декомпозиции обычно представляют в виде иерархической схемы, на нижнем уровне которой располагают сравнительно простые блоки, а на верхнем - систему, подлежащую разработке. На каждом иерархическом уровне описание блоков выполняют с определенной степенью детализации, абстрагируясь от несущественных деталей. Соответственно, для каждого уровня используют свои формы документации и свои модели, отражающие сущность процессов, выполняемых каждым блоком. Так для ПС в целом, как правило, удается сформулировать лишь самые общие требования, а блоки нижнего уровня должны быть специфицированы так, чтобы из них действительно можно было собрать работающую систему. Иными словами, чем больше блок, тем более абстрактным должно быть его описание.

Декомпозицию ПС на блоки в ходе пошаговой детализации можно выполнять по различным признакам. Например, классическая (структурно-модульная) технология программирования предполагает, что система разбивается по функциям (функциональная декомпозиция). Далее функции группируются в модули, которые независимо отлаживаются и используются для сборки результирующей системы. Объектная технология программирования предусматривает разбиение проектируемой системы по классам и объектам предметной области (объектная декомпозиция). Из множества полученных классов формируется классовая иерархия (одна или несколько), составляющая концептуальный каркас проектируемой системы. Из совокупности объектов строится так называемая объектная структура ПС, описывающая функционирование системы через поведение и взаимодействие объектов. Известна также декомпозиция ПС по обрабатываемым данным, в ходе которой осуществляется выделение и разбиение на составляющие основных потоков данных, обрабатываемых системой.

В данном лабораторном практикуме за основу берется метод пошаговой детализации на основе объектной декомпозиции. Предполагается, что студенты имеют базовые знания в области объектно-ориентированного и визуального программирования, владеют навыками работы в одной из сред быстрой разработки приложений, умеют составлять и отлаживать простые программы на языке C++.

3. Формулировка задания и этапы выполнения работ. Требования к программной системе

В ходе лабораторного практикума необходимо разработать модульную программную систему имитационного моделирования заданной системы массового обслуживания с использованием объектной декомпозиции. Этапы разработки ПС рассматриваются как отдельные лабораторные работы, в совокупности составляющие лабораторный практикум.

Этапы выполнения задания:

- получить вариант задания в соответствии с индивидуальным номером (номером по журналу);
- построить модель заданной системы массового обслуживания;
- разработать иерархию классов, а также модульную и объектную структуру проектируемой программной системы;
- реализовать разработанную иерархию, модульную и объектную структуру программной системы на заданном языке программирования.

Проектируемая ПС должна соответствовать следующим требованиям.

I Функциональные требования:

- диалоговый режим работы;
- отображение текущего состояния процесса моделирования с указанием модельного времени и наиболее важных параметров;
- возможность прерывания процесса моделирования на любом его этапе с распечаткой промежуточных результатов;
- отображение текущего состояния процесса моделирования в текстовом, табличном и/или графическом виде;
- отображение результатов моделирования в текстовом и/или табличном виде.

Результаты моделирования для всех вариантов задания включают: 1) число выработанных заявок; 2) число потерянных заявок; 3) загрузка всех устройств; 4) максимальные и средние длины всех очередей; 5) дополнительные результаты, определенные вариантом задания.

II Требования к условиям разработки и эксплуатации:

- IBM-совместимая ПЭВМ;
- операционная система семейства Windows (версии 7, 8 или новее);
- сообщения на русском языке;
- язык программирования C++;
- среда программирования Microsoft Visual Studio, Embarcadero RAD Studio Architect (Borland C++ Builder).

4. Пример выполнения задания

Порядок выполнения работ, входящих в лабораторный практикум, будем пояснять на примере проектирования программной системы, предназначенной для имитационного моделирования модуля матричной коммуникационной сети (КС).

Этап 1. Изучение формулировки задания.

Укрупненная структурная схема моделируемого модуля КС изображена на рисунке 1. Модуль содержит буферные блоки (ББ), блок опроса (БО), блок выбора направления передачи пакетов (БВНПП), буферный регистр Рг, мультиплексор МХ и демультиплексор DX.

Входы I_1, …, I_N используются для приема пакетов от соседних модулей КС. Вход I_0 служит для приема пакетов, вырабатываемых процессором, обслуживаемым данным модулем. Буферные блоки $ББ_0$, $ББ_1$, …, $ББ_N$ обеспечивают прием, хранение и выдачу пакетов в порядке их поступления на соответствующие входы. Блок опроса (БО) управляет считыванием сообщений из ББ через мультиплексор МХ в буферный регистр Рг и реализует требуемую дисциплину обслуживания ББ (в данном случае предполагается простейшая последовательная дисциплина опроса ББ). Блок выбора направления передачи пакетов (БВНПП) определяет направление выдачи каждого пакета согласно заданному алгоритму маршрутизации и обеспечивает выдачу пакета через демультиплексор DX на соответствующий этому направлению выход *O*.

Процедура обработки пакетов данным модулем включает последовательность из трех фаз.

Фаза 1. Прием пакетов от соседних модулей и запись в соответствующие ББ.

Фаза 2. Считывание пакетов из очередей в порядке поступления и определение направлений их выдачи.

Фаза 3. Выдача пакетов соседним модулям и процессору, который соответствует текущему модулю.

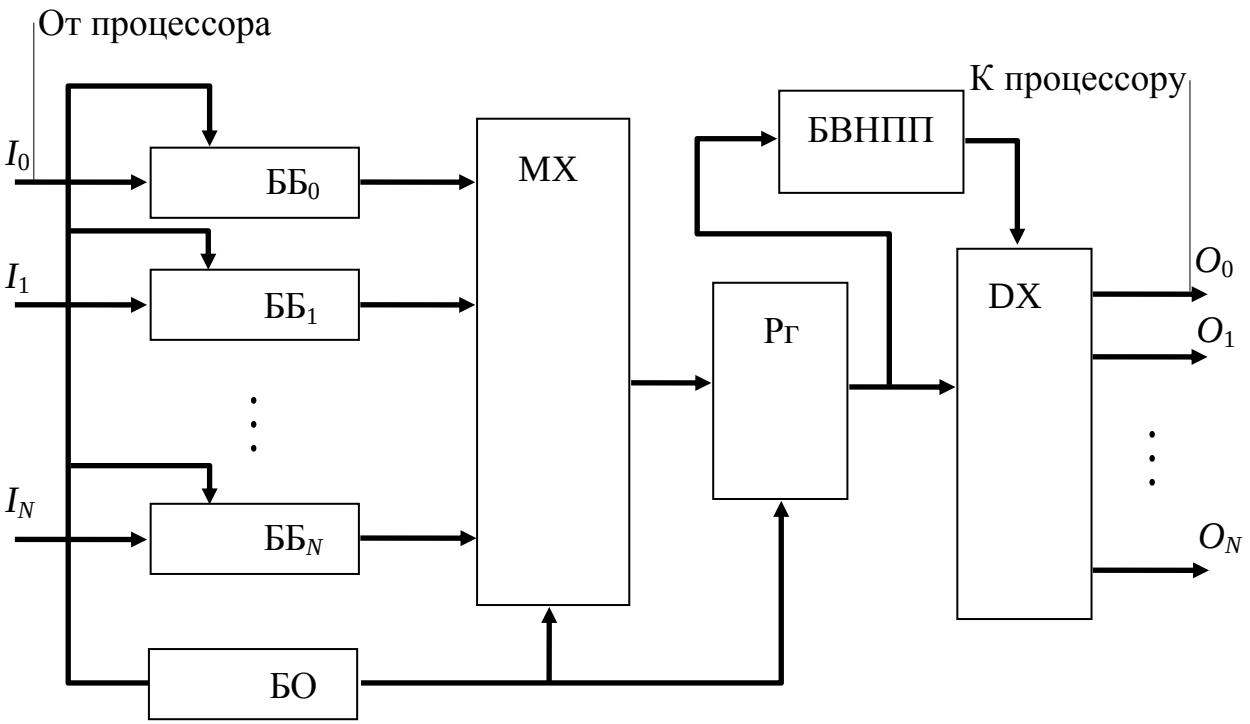

Рисунок 1. Структура модуля коммуникационной сети

Этап 2. Построение имитационной модели.

Для построения имитационной модели модуля КС будем использовать расширенный язык Q-схем. Он представляет собой полуформальный графический язык, который органично вписывается в рамки объектного подхода. Он универсален и позволяет описывать системы широкого класса.

Q-схема – это граф, вершины которого отображают элементы моделируемой системы, а дуги связывают элементы между собой. Все вершины разделяются на 4 вида в зависимости от того, какой элемент они моделируют. Связи делятся на 2 вида.

Графическое изображение основных элементов Q-схемы дано на рисунке 2. Кружком без входящих стрелок обозначается генератор заявок. Его функция – имитировать поступление заявок в сис-

тему извне в соответствии с некоторым законом распределения (в нашем случае заявкой будет пакет). Набор параметров λ задается в соответствии с законом распределения. Кружок с входящими стрелками справа от генератора – это канал. Его функция – имитация обработки заявки в течение времени, определяемого некоторым законом распределения времен обслуживания. Набор параметров μ аналогичен λ для генератора. Следующий элемент – прямоугольник с вертикальными линиями – накопитель (или очередь). Он имитирует возможные скопления заявок в системе. Накопитель характеризуется предельной длиной L. Она может быть бесконечной (накопитель без потерь) или ограниченной (накопитель с потерей заявок). Также накопитель характеризуется дисциплиной обслуживания заявок. Типовые дисциплины – FIFO (первым пришел, первым ушел) и LIFO (первым пришел, последним ушел), но возможны и другие, например, случайный выбор заявок. Наконец, четвертый элемент – треугольник – это клапан. Он имитирует управление потоками заявок. Например, он может блокировать прохождение заявок или, наоборот, разрешить в зависимости от значений некоторых управляющих сигналов, которые поступают от других элементов.

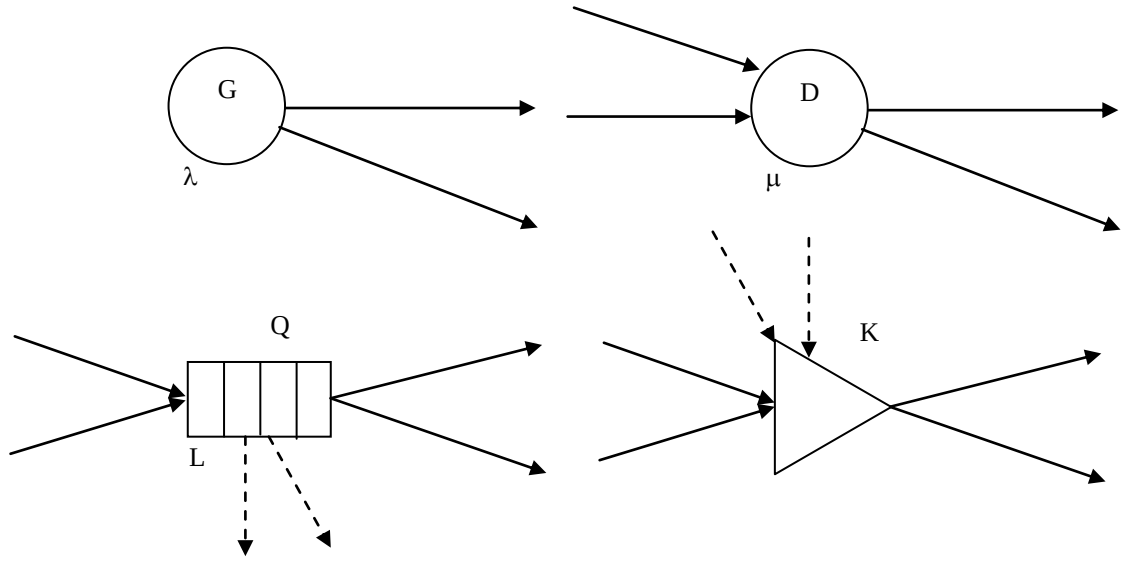

Рисунок 2. Основные элементы Q-схемы

Связи в Q-схеме подразделяются на 2 вида. Информационные связи – изображаются сплошными стрелками – отображают пути прохождения заявок в системе. Управляющие связи – рисуются пунктирными стрелками – показывают пути движения управляю-

щих сигналов в системе. На связи накладывается ряд синтаксических и семантических ограничений. Так, генератор (если он не является управляемым) не может иметь входящих связей, и обязан иметь хотя бы одну выходную информационную связь. А у клапанов, например, обязательно должны быть входящие управляющие связи. Выходящие управляющие связи допустимы только у каналов и очередей. Их число произвольно.

Каждой управляющей связи в Q-схеме соответствует определенное состояние управления, например, «канал пуст» или «очередь занята». Если в текущий момент модельного времени такое состояние имеется, то управляемый элемент воспринимает его как разрешающий сигнал. Например, клапан может открыться по условию «пустоты» очереди. Если некоторый элемент имеет более одной входящей управляющей связи, разрешающие сигналы, приписанные этим связям, комбинируются логической функцией. Это может быть конъюнкция, дизъюнкция или иные функции.

Для обеспечения компактности Q-схем базовый набор элементов дополняется вспомогательными элементами группового управления. Графическое изображение этих элементов дано на рисунке 3.

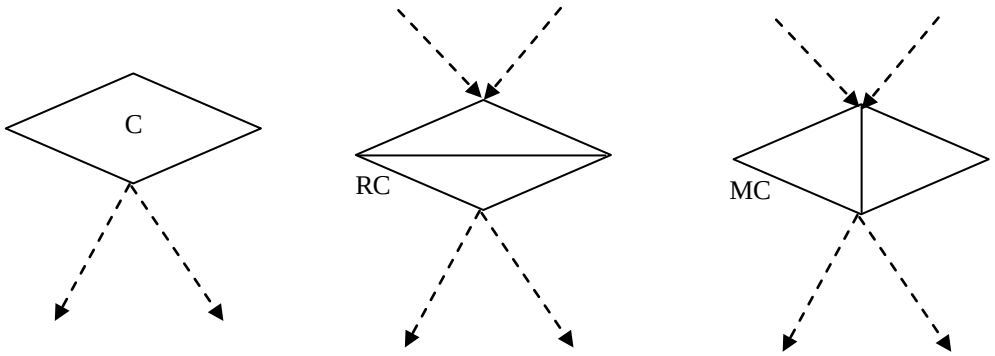

Рисунок 3. Вспомогательные элементы Q-схемы

Элемент слева, называемый простым случайным контроллером, реализует управление конечной группой клапанов по некоторому случайному закону. Он может, например, равновероятно открывать один из двух или более клапанов. Вероятности не обязательно должны совпадать. Элемент в середине – адаптивный случайный контроллер – также имитирует случайный выбор одного клапана из двух или более. Но он может автоматически менять ве-

роятности открытия клапанов в зависимости от состояния других элементов. Элемент справа – обобщенный массовый контроллер – тоже необходим для выбора клапанов из заданной группы. Однако он может выбирать клапаны по любому закону, не обязательно случайному (закон задается проектировщиком Q-схемы). Кроме того, он выбирает не один из N, а M из N клапанов, возможно, учитывая состояние других элементов схемы. Выбранные клапаны могут быть либо открыты, либо закрыты, что определяется реакцией клапанов на сигналы от элементов-контроллеров.

Итак, используя описанный язык Q-схем, построим имитационную модель модуля матричной КС, основываясь на содержательном описании его структуры и функционирования. Полученная модель (Q-схема) изображена на рисунке 4.

Назначение элементов разработанной модели состоит в следующем.

Генераторы MSGi имитируют входящие потоки пакетов.

Очереди BUFi соответствуют буферным блокам ББ.

Клапаны KBUFi предназначены для разрешения / запрета выбора пакетов из соответствующих буферных блоков для обработки.

Клапаны KLOSTi имитируют выход из системы потерянных пакетов при переполнении буферов.

Канал RG моделирует обработку очередного пакета, считанного из одного из ББ.

Генератор MXG служит для имитации тактов работы блока опроса.

Каналы MXBUFi обеспечивают имитацию выбора i-го буфера на 1 такт.

Каналы MXDELi предназначены для имитации обработки считанного пакета в течение 2 тактов.

Клапаны KMXON и KMXOFF имитируют управление включением и отключением блока опроса.

Клапан KSTART служит для пометки начала цикла опроса буферов, а клапан KSTOP – для пометки конца цикла опроса.

Фиктивная очередь TMPBUF моделирует хранение условия пуска цикла опроса (ее длина равна 1).

Через exp и nonrnd обозначены соответственно экспоненциальный и фиксированный (неслучайный) законы распределения времен появления заявок и обслуживаний; в скобках указан параметр закона распределения.

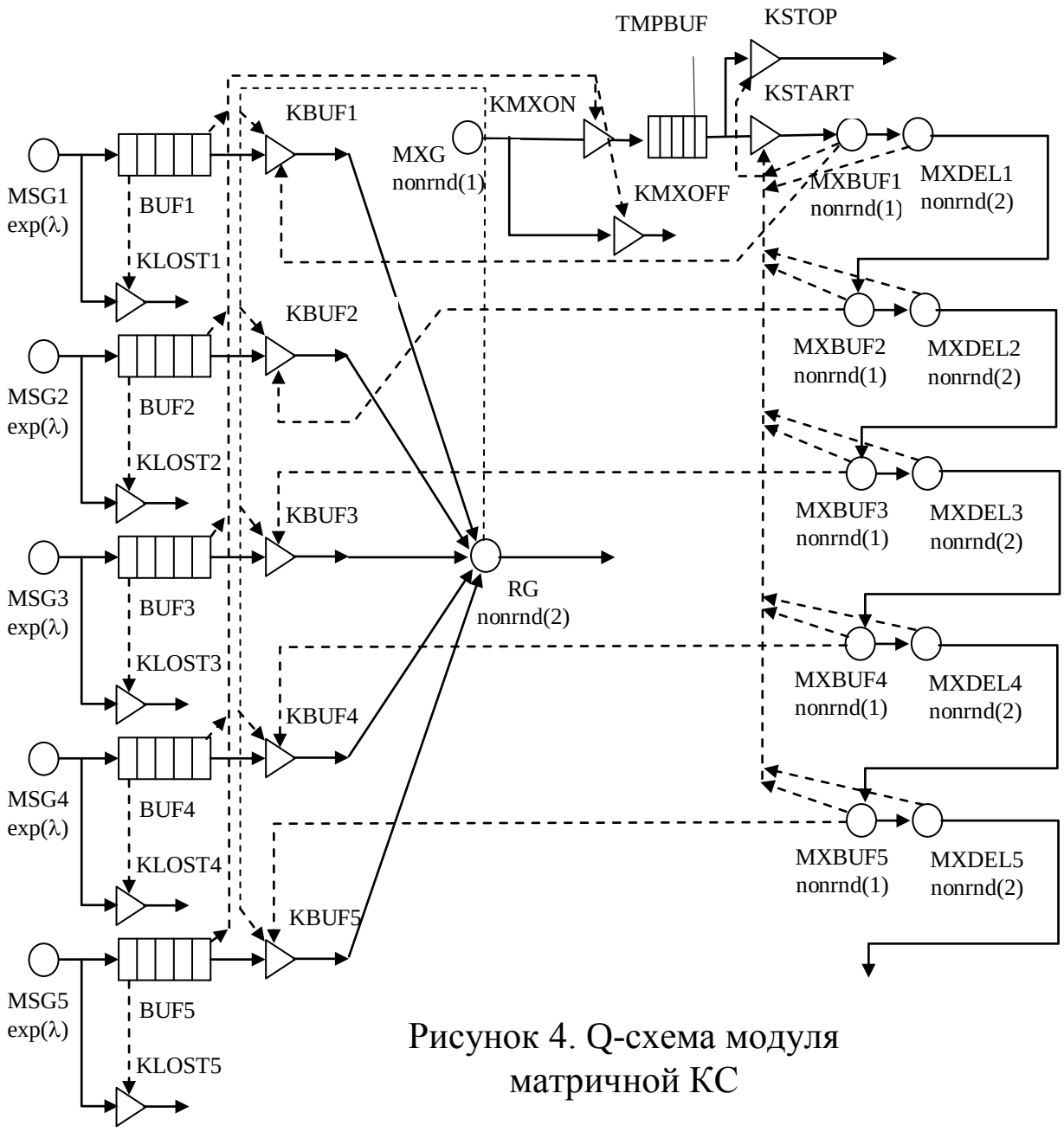

Рисунок 4. Q-схема модуля матричной КС

Открытие клапанов обеспечивается согласно следующим условиям:

KLOSTi:	BUFi полон;
KBUFi:	RG пуст И MXBUFi не пуст;
KMXON:	хотя бы один BUFi не пуст;
KMXOFF:	все BUFi пусты;
KSTART:	все MXBUFi и MXDELi пусты;
KSTOP:	хотя бы один MXBUFi или MXDELi не пуст.

Анализируя порядок прохождения заявок в разработанной Q-схеме при различных соотношениях ее параметров, несложно убедиться в адекватности этой модели представляемому объекту.

Этап 3. Построение иерархии классов проектируемой ПС.

Для построения универсальной и наращиваемой иерархии классов, представляющей Q-схемы широкого класса, необходимо выявление всех абстракций предметной области, их анализ, выделение общих свойств, поведения и отличий. По спискам общих свойств и отличий можно получить более общие абстракции и затем построить отношение «обобщения», которое будет основой иерархии классов.

Все выявленные абстракции вместе с их характеристиками сведены в таблицу 1.

Таблица 1

№	Название абстракции	Характеристика абстракции
1	Генератор заявок	Входит в структуру Q-схемы Обрабатывает заявки Имеет выходные информационные связи Выдает заявки Задерживает заявки (по времени) Характеризуется законом распределения Не имеет входов Не имеет управляющих выходов
2	Канал	Входит в структуру Q-схемы Обрабатывает заявки Имеет входные информационные связи Имеет выходные информационные связи Принимает заявки Выдает заявки Задерживает заявки (по времени) Характеризуется законом распределения Не имеет управляющих входов Может иметь управляющие выходы
3	Очередь	Входит в структуру Q-схемы Обрабатывает заявки Имеет входные информационные связи Имеет выходные информационные связи Принимает заявки Выдает заявки Задерживает заявки (по готовности приемника) Характеризуется дисциплиной обслуживания Имеет контейнер заявок Не имеет управляющих входов Может иметь управляющие выходы

4	Клапан	Входит в структуру Q-схемы Обрабатывает заявки Имеет входные информационные связи Имеет входные управляющие связи Имеет выходные информационные связи Принимает заявки Выдает заявки Не задерживает заявки Характеризуется функцией управления Не имеет управляющих выходов
5	Простой случайный контроллер	Входит в структуру Q-схемы Не обрабатывает заявки Имеет хотя бы 2 управляющих выхода Управляет группой клапанов по случайному закону Выбор клапанов по принципу «один из N» Не имеет информационных связей Характеризуется законом распределения Не имеет управляющих входов
6	Адаптивный случайный контроллер	Входит в структуру Q-схемы Не обрабатывает заявки Имеет хотя бы 2 управляющих выхода Управляет группой клапанов по случайному закону Выбор клапанов по принципу «один из N» Адаптирует закон управления по состоянию других элементов Не имеет информационных связей Характеризуется законом распределения Имеет управляющие входы (ассоциации)
7	Обобщенный массовый контроллер	Входит в структуру Q-схемы Не обрабатывает заявки Имеет хотя бы 2 управляющих выхода Управляет группой клапанов по произвольному закону Выбор клапанов по любому принципу Адаптирует закон управления по состоянию других элементов и другой информации Не имеет информационных связей Имеет управляющие входы (ассоциации)
8	Заявка	Формируется в генераторе Включает атрибуты (тип, приоритет, время генерации) Уничтожается при выходе из системы
9	Структура схемы	Содержит информационные и управляющие связи для всех пар связанных элементов Для управляющих связей задает состояния управления
10	Q-схема	Охватывает управление запуском модели, остановом моделирования, организует сбор статистики, имитирует попарную передачу заявок элементами, управляет отслеживанием ошибок
11	Генератор случайных чисел	Используется генераторами заявок и каналами для получения временных интервалов выдачи заявок и обслуживания
12	Дисциплина обслуживания	Используется очередями для задания порядка извлечения заявок

13	Функция управления	Используется клапанами для объединения сигналов управления
14	Менеджер статистики	Вспомогательная абстракция, используемая Q-схемой для управления сбором статистики и ведения протокола статистики
15	Менеджер ошибок	Вспомогательная абстракция, используемая Q-схемой для управления отслеживанием ошибок

Из таблицы 1 можно увидеть, что многие элементы Q-схем имеют общие свойства и поведение, что позволяет построить на их основе обобщенные абстракции. В дальнейшем эти абстракции могут быть представлены абстрактными классами, которые, в свою очередь, будут порождать неабстрактные классы элементов Q-схем. Список выявленных новых абстракций представлен в таблице 2.

Таблица 2

№	Название абстракции	Характеристика абстракции
1	Структурный элемент Q-схемы	Входит в структуру Q-схемы Имеет какие-либо связи (характер связей не определен)
2	Обработчик заявок	Входит в структуру Q-схемы Обрабатывает заявки Имеет информационные связи (входные или выходные, или входные и выходные одновременно)
3	Контроллер группы клапанов	Входит в структуру Q-схемы Не обрабатывает заявок Имеет управляющие связи (характер связей неизвестен) Не имеет информационных связей
4	Передатчик	Входит в структуру Q-схемы Обрабатывает заявки Имеет выходные информационные связи Выдает заявки (с задержкой либо без нее)
5	Приемник	Входит в структуру Q-схемы Обрабатывает заявки Имеет входные информационные связи Принимает заявки
6	Передатчик с задержкой	Входит в структуру Q-схемы Обрабатывает заявки Имеет выходные информационные связи Выдает заявки Задерживает заявки (по времени в соответствии с заданным законом распределения)
7	Объект управления	Входит в структуру Q-схемы Обрабатывает заявки Имеет входные управляющие связи Характеризуется функцией управления

Далее для получения иерархии классов будем последовательно устанавливать отношение «общее – частное» для всех абстракций, сведенных в таблицы 1 и 2, а также определять отношения использования и ассоциации. Иерархия классов представлена на рисунке 5. Прямоугольниками отображены основные элементы Q-схем (которые непосредственно изображаются на схеме), а овалами показаны дополнительные абстракции. Сплошными стрелками изображено отношение открытого наследования, которое представляет в иерархии отношение «общее – частное», а пунктиром показаны ассоциации и отношение использования между абстракциями.

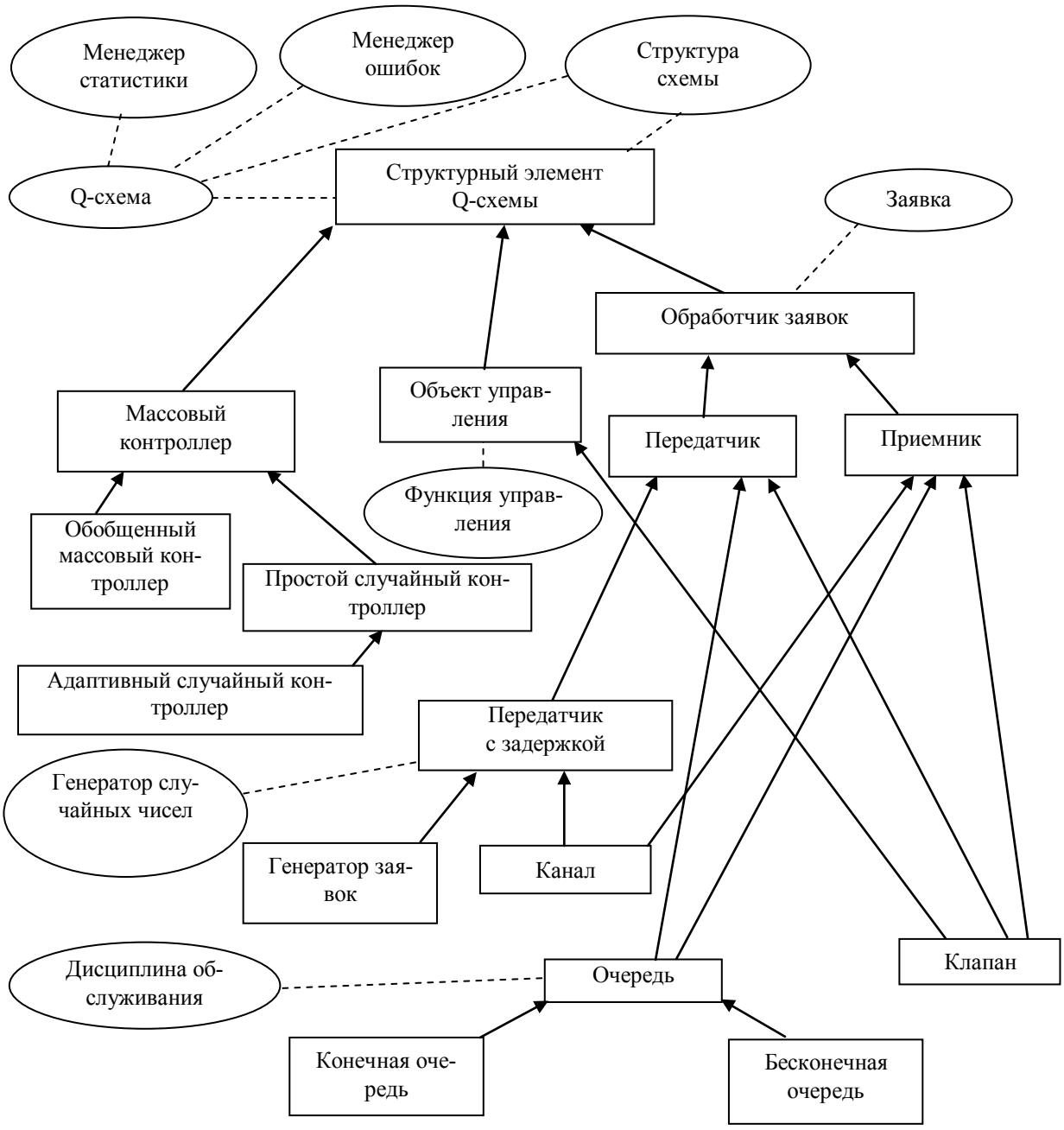

Рисунок 5. Иерархия классов

Каждая из абстракций разработанной иерархии классов далее представляется в виде сущности одного из трех видов: класса (структуры), шаблонного класса или функции на языке C++. Переход от абстракций к сущностям на C++ осуществляется на следующем этапе. Ниже в таблице 3 записано соответствие сущностей C++ выделенным абстракциям и даны имена (идентификаторы) сущностей.

Таблица 3

Наименование абстракции	Вид сущности	Идентификатор сущности
Структурный элемент Q-схемы	класс	CGenericElement
Обработчик заявок	класс	CReqAwareElement
Объект управления	класс	CObjectOfControl
Приемник	класс	CReceiver
Передатчик	класс	CTransmitter
Передатчик с задержкой	шаблон	CDelayedTransmitter
Неуправляемый генератор заявок	шаблон	CReqGenerator
Управляемый генератор заявок	шаблон	CControlledReqGenerator
Канал	шаблон	CDevice
Клапан	класс	CGate
Абстрактная очередь	шаблон	CQueue
Бесконечная очередь	шаблон	CUnlimitedQueue
Конечная очередь	шаблон	CLimitedQueue
Контроллер группы клапанов	класс	CMassGateController
Простой случайный контроллер	класс	CRndGateController
Адаптивный случайный контроллер	класс	CSmartRndGateController
Обобщенный массовый контроллер	класс	CCustomMassGateController
Q-схема	класс (с единственным объектом)	CQChart
Заявка	класс	CReq
Структура Q-схемы	набор структур	–
Генератор случайных чисел	шаблон	Определяется пользователем
Дисциплина обслуживания	шаблон	Определяется пользователем
Функция управления	callback-функция	Определяется пользователем
Менеджер статистики	класс (с единственным объектом)	CStatManager
Менеджер ошибок	класс (с единственным объектом)	CErrorManager

Этап 4. Построение модульной и объектной структуры проектируемой ПС.

Структура ПС должна обеспечивать удобство ее использования при составлении проектов моделирования с различными пара-

метрами. При этом она должна предоставлять возможность внедрения внешнего кода в контекст сущностей с целью обеспечения открытости и гибкости. Так, необходима возможность добавления определенных пользователем классов генераторов случайных чисел для реализации специальных законов распределения генераторов заявок и каналов, введения новых функций управления клапанами и дисциплин обслуживания в очередях. Это же касается и обобщенных массовых контроллеров, которые по определению должны включать код пользователя, представляющий закон управления клапанами.

Состав основных модулей и заголовочных файлов программной системы вместе с описанием их назначения представлены в таблице 4.

Таблица 4

№№	Имя модуля	Назначение модуля
1	qchartelements.h qchartelements.cpp	Содержит определения и реализации классов основных элементов Q-схемы
2	qchart.h qchart.cpp	Включает определение и реализацию классов CQChart и CStatManager, а также дополнительных классов для поддержки цикла моделирования
3	qchartexcept.h qchartexcept.cpp	Содержит определение и реализацию класса CErrorManager и иерархию классов исключительных ситуаций для представления ошибок
4	qchartgatectrl.h qchartgatectrl.cpp	Определяет ряд базовых функций управления клапанами (например, И, ИЛИ, И-НЕ, ИЛИ-НЕ)
5	qchartstdhdr.h	Стандартный заголовочный файл, обеспечивающий компоновку проектов и содержащий макросы для упрощения интерфейса библиотеки
6	qchartcommon.h	Описывает базовые типы библиотеки
7	qchartrndgen.h	Содержит определения шаблонных классов типовых генераторов случайных чисел (фиксированный, равномерный закон, экспоненциальный)
8	qchartqdisc.h	Определяет классы для реализации стандартных дисциплин обслуживания (FIFO, LIFO)

Модульная структура программной системы представлена на рисунке 6.

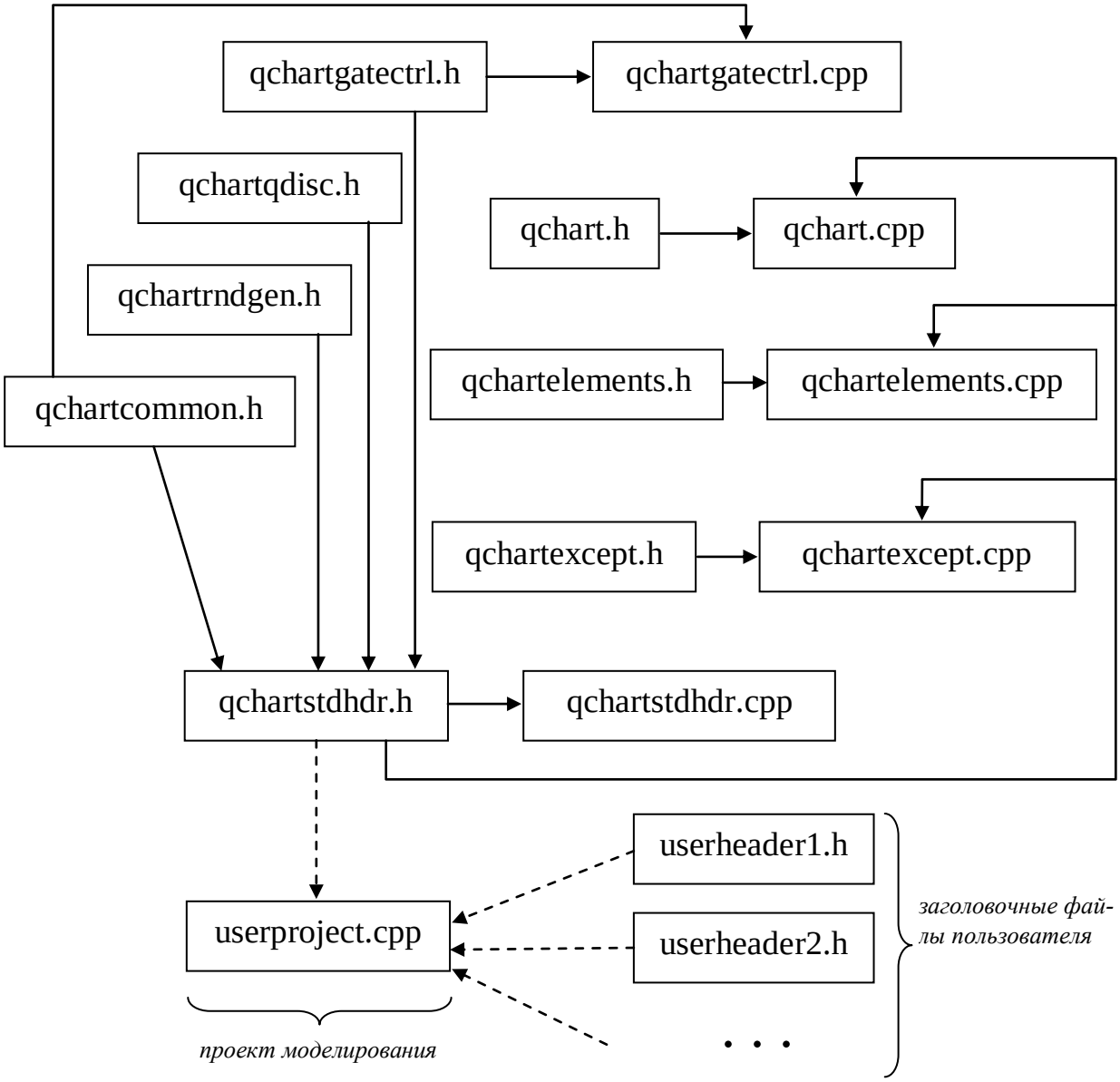

Рисунок 6. Модульная структура
ПС имитационного моделирования

Этап 5. Разработка содержания модулей и классов проектируемой ПС.

Первоначально определим организацию моделирующего проекта.

Будем представлять моделирующий проект в виде отдельного приложения, использующего разработанную модульную структуру ПС. Пользователь должен подготовить основной модуль проекта (cpp-файл userproject.cpp), вписав в него в соответствии с требуемой структурой определения элементов, связей, параметров моделирования (согласно варианту задания) и запустив процесс модели-

рования соответствующей функцией. Структуру этого модуля можно представить, например, следующим образом:

```
// подключение файлов ПС
#include "qchartstdhdr.h"
...
using namespace QChart;
// пользовательские заголовочные файлы
...
// основная функция моделирующего проекта
...
      // определение и регистрация генераторов
      ...
      // определение и регистрация каналов
      ...
      // определение и регистрация очередей
      ...
      // определение и регистрация клапанов
      ...
      // регистрация информационных связей
      ...
      // регистрация управляющих связей
      ...
      // настройка параметров моделирования
      ...
      // пуск моделирования
      ...
```

Определим содержание модулей ПС и классов, входящих в построенную иерархию. Содержание классов будем разрабатывать исходя из характеристик абстракций, сведенных в таблицы 1 и 2. Распределение классов по модулям будем осуществлять с учетом данных из таблицы 4. Первоначально будем определять базовые классы, на основе которых затем определим все производные классы и шаблоны. Параллельно будем записывать определения вспомогательных классов, а также дополнительных структур данных, типов и функций.

Ниже в качестве примера дано определение класса CGenericElement, представляющего абстрактный структурный элемент Q-схемы:

```
class CGenericElement {
    // абстрактный элемент Q-схемы
    // инкапсулирует идентификацию элементов и общие структурные свойства
  public:
      // конструктор и деструктор
    explicit __fastcall CGenericElement(const std::string & name):
        _plinks(0), __name(name), __id(0)
    {
        ;
    }
```

```
    virtual ~CGenericElement() = 0;

    // общая информация об элементе
    const std::string & GetName() const { return __name; }
    int GetID() const {    return __id; }

  protected:
    ChartStrEntry_t * _plinks;
        // указатель на ячейку структуры Q-схемы с описанием связей

  private:
    std::string __name; // символическое имя элемента
    int __id;           // числовой идентификатор элемента

    // друзья
    friend class CQChart; // для настройки связей с Q-схемой

    // избыточные функции
    CGenericElement(const CGenericElement &);
    CGenericElement & operator = (const CGenericElement &);

}; // CGenericElement
```

В приведенном выше определении класса использовался тип ChartStrEntry_t, представляющий отдельную ячейку структуры Q-схемы. Его определение имеет следующий вид:

```
// тип элемента списка связей Q-схемы
typedef struct LinksListEntry_t {
    CGenericElement  * pelem;       // указатель на элемент
    LinksListEntry_t * pnextlink;   // указатель на следующую связь
} LinksListEntry_tag;

// тип ячейки структуры Q-схемы
typedef struct ChartStrEntry_t {
    CGenericElement  * pelem;           // текущий элемент
    LinksListEntry_t * pininflinks;     // входные информационные связи
    LinksListEntry_t * poutinflinks;    // выходные информационные связи
    LinksListEntry_t * pinctrllinks;    // входные управляющие связи
    LinksListEntry_t * poutctrllinks;   // выходные управляющие связи
    ChartStrEntry_t  * pnextelem;       // указатель на следующую ячейку
} ChartStrEntry_tag;
```

Как видно из последнего определения, структура Q-схемы задается односвязным списком ячеек. В каждой из них имеются указатели на соответствующий элемент Q-схемы, на его всевозможные связи с другими элементами, а также на очередную аналогичную ячейку, описывающую следующий элемент. Каждая связь определяется структурой LinksListEntry_t.

Аналогичным образом определяем остальные классы, двигая вниз по иерархии (от общего к частному).

Одним из важнейших классов иерархии является CReqAwareElement – абстрактный обработчик заявок. Этот класс

определяет базовые аспекты поведения всех обработчиков заявок. Его определение дано ниже:

```
class CReqAwareElement: virtual public CGenericElement {
    // абстрактный обработчик заявок
    // от этого класса наследуют все классы, работающие с заявками
  public:
      // конструктор и деструктор
    explicit __fastcall CReqAwareElement(
        const std::string & name, bool logstat = true): CGenericElement(name)
    {
        __logstat = logstat;
        _marksreqprocessed = false;
    }
    virtual ~CReqAwareElement() { }

    // признак выдачи статистики для элемента
    bool GetLogStat() const { return __logstat; }
    // установка признака пометки обработанных заявок
    void SetMarksReqProcessed() { _marksreqprocessed = true; }
    // получение состояния - для управления другими элементами
    virtual state_t GetState() const = 0;
    // механизм вычисления состояния определяется подклассами

  protected:
    class CReq { // класс "заявка"
        // вложенность класса моделирует ассоциацию элементов с заявками
      public:
        // конструктор
        explicit __fastcall CReq(bool logstat = true) {
            __inittimestamp = __timestamp = 0;
            __logstat = logstat;
            __processed = false;
        }
        // информация
        size_t GetTimeStamp() const { // время пребывания заявки в системе
            return __timestamp;
        }
        size_t GetInitTimeStamp() const { // время появления заявки в системе
            return __inittimestamp;
        }
        bool GetLogStat() const { // признак включения заявки в статистику
            return __logstat;
        }
        bool GetProcessed() const { // признак обработки заявки
            return __processed;
        }
        // счет времени пребывания заявки в системе
        void __fastcall SetTimeStamp(size_t time) { __timestamp = time; }
        void __fastcall SetInitTimeStamp(size_t time)
        { __inittimestamp = time; }
        // пометка обработанной заявки
        void SetProcessed() { __processed = true; }

      private:
        size_t __inittimestamp; // время появления заявки в системе
        size_t __timestamp; // текущая временная метка
            // используется для подсчета времени пребывания заявки в системе
        bool __logstat; // признак включения текущей заявки в статистику
        bool __processed; // признак попадания заявки хотя бы в один канал
        // если __processed == false, то заявка еще не обрабатывалась
```

```
        // избыточные функции
        CReq & operator = (const CReq &);

    }; // CReq

    bool _marksreqprocessed; // признак пометки обработанных заявок

  private:
    bool __logstat; // признак записи статистики по заявкам текущего элемента

    // друзья
    friend struct ReqListEntry_t; // для доступа к заявке из очереди

    // избыточные функции
    CReqAwareElement(const CReqAwareElement &);
    CReqAwareElement & operator = (const CReqAwareElement &);

}; // CReqAwareElement
```

Особенностью представленного выше класса является то, что непосредственно внутри него определяется вложенный класс CReq, описывающий заявку. Вложение классов в данном случае отображает отношение ассоциации (заявки ассоциируются со всеми элементами, которые их обрабатывают). Внутри класса CReq заданы атрибуты заявок, например, время поступления заявки в систему, признак прохождения заявкой хотя бы одного обработчика. При необходимости перечень этих атрибутов можно расширить. Для организации доступа к свойствам заявки заданы соответствующие компонентные функции. Некоторые свойства доступны только для чтения, некоторые доступны и для модификации.

Возвращаясь к классу CReqAwareElement, следует отметить, что в нем введена чисто виртуальная функция GetState, позволяющая получить текущее состояние элемента. Поскольку способ получения состояния элемента и число возможных состояний зависят от вида этого элемента, данная функция объявлена абстрактной и будет определяться производными классами.

На основе класса CReqAwareElement непосредственно определяется класс CTransmitter, представляющий все элементы, имеющие выходящее информационные связи и способные выдавать заявки. Определение класса CTransmitter будет следующим:

```
class CTransmitter: virtual public CReqAwareElement {
      // абстрактный элемент Q-схемы, способный выдавать заявки
  public:
      // конструктор и деструктор
      explicit __fastcall CTransmitter(const std::string & name):
        CGenericElement(name), CReqAwareElement(name)
      {
        ;
```

```
    }
    ~CTransmitter() { }

  protected:
    // поведение
    virtual CReq * __fastcall _Transmit(CReq * req = 0) = 0; // выдача заявки
    // информация
    virtual bool _ReadyToTransmit() const = 0; // готовность к выдаче заявки

  private:
    // друзья
    friend class CQChart;

    // избыточные функции
    CTransmitter(const CGenericElement &);
    CTransmitter & operator = (const CGenericElement &);

}; // CTransmitter
```

В классе CTransmitter появляются две важные функции – _ReadyToTransmit и _Transmit. Первая служит для проверки готовности элемента к выдаче заявки, вторая – непосредственно для имитации выдачи заявки. Обе функции абстрактны, так оба моделируемые ими аспекта поведения зависят от вида обработчика заявок.

Класс CTransmitter порождает производный класс CDelayedTransmitter, описывающих элементы, которые передают элементы после некоторой задержки (характер задержки на данном уровне иерархии не определен):

```
class CDelayedTransmitter: public CTransmitter {
      // абстрактный элемент Q-схемы, выдающий заявки с задержкой
  public:
      // конструктор и деструктор
    explicit __fastcall CDelayedTransmitter(const std::string & name):
        CGenericElement(name), CReqAwareElement(name), CTransmitter(name),
        _preq(0), _issuetime(0), _issuetimeset(false)
    {
        ;
    }
    ~CDelayedTransmitter() {
        ;
        // элементы НЕ удаляют заявок; это делает фиктивный элемент
        // "сборщик обслуженных заявок"
    }

    // информация
    virtual state_t GetState() const { // проверка состояния
        return _preq == 0 ? empty : busy;
    }

  protected:
    // поведение
    virtual void _GenerateIssueTime() = 0; // время выхода заявки

    CReq * _preq; // заявка, находящаяся на обработке
```

```
    size_t _issuetime; // время выдачи заявки
    bool _issuetimeset; // признак, что время выдачи задано

  private:
    // друзья
    friend class CQChart;

    // избыточные функции
    CDelayedTransmitter(const CGenericElement &);
    CDelayedTransmitter & operator = (const CGenericElement &);

}; // CDelayedTransmitter
```

Особенностью этого класса является то, что в нем определяется унаследованная чисто виртуальная функция GetState. В зависимости от того, есть ли в данном элементе заявка в текущий момент модельного времени (_preq == 0 или _preq != 0), функция возвращает empty (элемент пуст) или busy (элемент занят).

Конкретизация содержания остальных классов иерархии осуществляется аналогично описанному. По мере определения классов конкретизируется реализация их компонентных функций. Для обычных классов определения функций размещаются в соответствующем cpp-файле qchartelements.cpp, для шаблонных классов все определения остаются в заголовочном файле qchartelements.h.

Этап 6. Определение объектной структуры ПС. Запись определения Q-схемы в моделирующем проекте.

На данном этапе требуется задать Q-схему моделируемой системы массового обслуживания (рисунок 4) в основном файле моделирующего проекта, а также определить общие параметры моделирования (условие окончания моделирования, условие начала сбора статистических данных, перечень элементов, данные о которых необходимо получить и т.д.).

Для задания Q-схемы необходимо создать множество объектов, представляющих все ее элементы, и зарегистрировать эти объекты и связи между ними в структуре данных ChartStrEntry_t, описанной выше. Регистрация элементов и связей осуществляется с помощью компонентных функций класса CQChart, представляющего Q-схему, и сводится к добавлению указателей на объекты-элементы и ячейки описания связей, содержащие атрибуты связей. Функция RegisterElement регистрирует новый элемент, а функция RegisterLink регистрирует новую связь. Эти функции предполагают передачу большого числа аргументов, состав которых зависит от

типа регистрируемого элемента или связи. В связи с этим доступ к этим функциям удобно организовать через вспомогательные макроопределения (макросы регистрации), сокращающие запись вызова функций.

Ниже даны заголовки некоторых макросов регистрации элементов Q-схем:

```
GEXP(latency,name,title,logstat)
GEVEN(lbound,ubound,name,title,logstat)
GNRND(latency,name,title,logstat)
DEXP(latency,name,title,logstat)
DEVEN(lbound,ubound,name,title,logstat)
DNRND(latency,name,title,logstat)
FIFOLQ(name,title,len,logstat)
FIFOUQ(name,title,logstat)
XXXGT(name,title)
RNDCTRL(name,title);
SMRNDCTRL(name,title);
MASSUSERCTRL(id,title,assoccontrollers,ctrlfunc);
```

Каждый из перечисленных макросов решает две задачи: вводит определение элемента Q-схемы в контекст проекта и регистрирует элемент. Названия макросов соответствуют задаваемым ими элементам. GXXXX задает неуправляемый генератор заявок (буква G) с законом распределения XXXX, DXXXX задает канал (буква D) с законом распределения XXXX, FIFOXQ определяет FIFO-очередь с ограничением по длине (X = L) или без ограничения (X = U), XXXGT определяет клапан с функцией объединения управляющих сигналов XXX (AND, OR, NOR и т.п.). Простой случайный контроллер определяется и регистрируется макросом RNDCTRL, адаптивный – макросом SMRNDCTRL, обобщенный массовый контроллер – макросом MASSUSERCTRL.

Назначение параметров макросов регистрации следующее: name – программный идентификатор элемента; title – символическое имя элемента (строка, отображаемая на экране); logstat – признак включения элемента в сбор статистических данных (true – включается, false – нет); latency, lbound, ubound – параметры законов распределения; len – длина очереди; assoccontrollers – вектор указателей на ассоциированные контроллеры (элементы, состояние которых учитывается массовым контроллером при выработке управляющих воздействий); ctrlfunc – указатель на пользовательскую функцию управления.

Ниже приведены примеры использования макросов регистрации элементов:

```
DNRND(2,RG,"Buffering register",true);
GEXP(21,MSG1,"Message generator 1",true);
FIFOLQ(TMPBUF,"Poling unit temporary queue",1,false);
NANDGT(KMXON,"Poling unit start gate");
```

Регистрация связей Q-схемы выполняется следующими макросами:

```
REG_ILINK
REG_CLINK
REG_CLINK_EX
REG_CLINK_RND
REG_CLINK_RND_EX
REG_CLINK_CUSTOM
```

Макрос REG_ILINK регистрирует информационную связь от элемента E1 к элементу E2. Формат его вызова:

```
REG_ILINK(E1,E2)
```

Остальные макросы используются для регистрации различных видов управляющих связей. Простейший из них – REG_CLINK – регистрирует связь от элемента E1 к управляемому элементу E2, устанавливая состояние управления по умолчанию full («очередь полна»). Формат его вызова таков:

```
REG_CLINK(E1,E2)
```

Макрос REG_CLINK_EX аналогичен, но позволяет задать любое допустимое для данного элемента-контроллера состояние управления (оно задается третьим аргументом), например:

```
REG_CLINK_EX(D1,K1,empty)
```

Макросы REG_CLINK_RND и REG_CLINK_RND_EX регистрируют управляющие связи от случайных контроллеров: первый – от простого контроллера, второй – от адаптивного контроллера. Форматы их вызова следующие:

```
REG_CLINK_RND(E1,E2,probability);
REG_CLINK_RND_EX(E1,E2,probability,assocelem,assocelemstate);
```

где probability – вероятность открытия клапана E2; assocelem – идентификатор ассоциированного контроллера; assocelemstate –

требуемое состояние ассоциированного контроллера (при котором клапан будет добавляться в группу управляемых элементов данного массового контроллера). В качестве assocelemstate можно использовать состояния empty, busy и (для очередей) full, empty_or_busy, empty_or_full, busy_or_full.

Под вероятностью открытия клапана (или иного управляемого элемента) здесь понимается вероятность его открытия в некотором такте модельного времени. Каждое значение вероятности – константа из диапазона (0…1). Сумма вероятностей открытия для группы клапанов, управляемых одним контроллером, должна равняться единице (условие полноты группы).

Поясним назначение и особенности использования ассоциированных контроллеров. Ассоциированный контроллер – это элемент, состояние которого определяет открытие/закрытие сопоставленного ему клапана из группы управляемых клапанов. Зная требуемое состояние ассоциированного контроллера, адаптивный контроллер может принять решение о включении или удалении клапана из группы управления. Для каждого клапана из группы управления указывается собственный ассоциированный контроллер. Им может быть любой элемент, способный принимать/передавать заявки и сообщать о своем состоянии путем выдачи управляющих сигналов. Таким образом, это может быть либо канал, либо очередь. Ассоциированный контроллер должен указываться при регистрации управляющей связи от адаптивного контроллера к соответствующему клапану. Одновременно должно задаваться и разрешенное состояние ассоциированного контроллера.

Завершающей стадией данного этапа проектирования программной системы является настройка параметров моделирования. Данная задача решается с помощью компонентных функций класса CQChart, описывающего Q-схему моделируемой системы. Как и в случае функций регистрации, рассмотренных выше, доступ к функциям настройки параметров моделирования удобно осуществлять через семейство макроопределений (макросов моделирования).

Макросы моделирования делятся на несколько групп в зависимости от задаваемого ими режима моделирования. Данный режим включает способ сбора статистики моделирования (запись в файл или вывод на экран), условие пуска цикла сбора статистики (с момента запуска моделирования, по выходу первой или N-й заявки,

по времени и т.п.), условие останова моделирования (по числу заявок, по времени моделирования и т.п.).

Список основных макросов моделирования, определенных в ходе проектирования ПС, приведен ниже:

SIMUL_STAT_COUT	моделирование с выдачей статистики на экран с запуском сбора статистики с начала моделирования и остановом по числу сгенерированных заявок
SIMUL_STAT_COUT_ON1ST	то же, но со сбором статистики с момента генерации первой заявки
SIMUL_STAT_COUT_ONNTH	то же, но со сбором статистики с момента генерации N-й заявки
SIMUL_STAT_COUT_ONTM	то же, но со сбором статистики с заданного момента модельного времени
SIMUL_STAT_FILE	моделирование с выдачей статистики в файл с запуском сбора статистики с начала моделирования и остановом по числу сгенерированных заявок
SIMUL_STAT_FILE_ON1ST	то же, но со сбором статистики с момента генерации первой заявки
SIMUL_STAT_FILE_ONNTH	то же, но со сбором статистики с момента генерации N-й заявки
SIMUL_STAT_FILE_ONTM	то же, но со сбором статистики с заданного момента модельного времени

Заголовки приведенных макросов имеют вид:

```
SIMUL_STAT_COUT(whenend, param)
SIMUL_STAT_COUT_ON1ST(whenend, param)
SIMUL_STAT_COUT_ONNTH(n, whenend, param)
SIMUL_STAT_COUT_ONTM(tm, whenend, param)
SIMUL_STAT_FILE(F, fwrmode, whenend, param)
SIMUL_STAT_FILE_ON1ST(F, fwrmode, whenend, param)
SIMUL_STAT_FILE_ONNTH(n, F, fwrmode, whenend, param)
SIMUL_STAT_FILE_ONTM(tm, F, fwrmode, whenend, param)
```

где whenend – способ завершения моделирования (возможные значения: duration – по времени моделирования; reqissued – по числу сгенерированных заявок; reqprocessed – по числу обработанных заявок; reqlost – по числу потерянных заявок); param – параметр длительности моделирования, его интерпретация зависит от значения whenend (например, при whenend = duration param – длительность цикла моделирования в единицах модельного времени, а при whenend = reqlost param – номер последней потерянной заявки); n – номер заявки, при генерации которой начинается сбор статистических данных; tm – момент модельного времени, с которого начинается сбор статистических данных; F – полное или сокращенное имя файла для хранения статистических данных; fwrmode – режим об-

новления файла статистики (значения: true – статистика добавляется в существующий файл; false – содержимое файла заменяется на новое).

Приведем примеры использования рассмотренных макросов моделирования.

Пример 1. Моделирование со сбором статистики с начала цикла моделирования с записью статистики в файл "qchart.stt" в текущем каталоге с заменой файла статистики, завершение по числу сгенерированных заявок (их число 400):

```
SIMUL_STAT_FILE("qchart.stt", false, reqissued, 400)
```

Пример 2. Моделирование со сбором статистики с момента времени 3000 с выводом статистики на экран с завершением в момент потери 200 заявок:

```
SIMUL_STAT_COUT_ONTM(3000, reqlost, 200)
```

Пример 3. Моделирование со сбором статистики с момента генерации 200-й заявки с записью статистики в файл "qchart.stt" с добавлением в конец файла и завершением в момент генерации 600-й заявки:

```
SIMUL_STAT_FILE_ONNTH(200, "qchart.stt", true, reqissued, 600)
```

Ниже представлен листинг проекта моделирования системы массового обслуживания, определенной вариантом задания.

```
//---------------------------------------------------------------------
// стандартные заголовки
#include "QChartStdHdr.h"

//---------------------------------------------------------------------
// заголовки пользователя

using namespace QChart;

// создание элементов Q-схемы

// генераторы
CReqGenerator<CNonRndNumGenerator<1> > MXG("Selector unit generator",false);
CReqGenerator<CExpRndNumGenerator<12> > MSG1("Message generator 1");
CReqGenerator<CExpRndNumGenerator<12> > MSG2("Message generator 2");
CReqGenerator<CExpRndNumGenerator<12> > MSG3("Message generator 3");
CReqGenerator<CExpRndNumGenerator<12> > MSG4("Message generator 4");
CReqGenerator<CExpRndNumGenerator<12> > MSG5("Message generator 5");

// каналы
```

```
CDevice<CNonRndNumGenerator<2> > RG("Buffering register");
CDevice<CNonRndNumGenerator<1> > MXBUF1("Queue 1 selector",false);
CDevice<CNonRndNumGenerator<1> > MXBUF2("Queue 2 selector",false);
CDevice<CNonRndNumGenerator<1> > MXBUF3("Queue 3 selector",false);
CDevice<CNonRndNumGenerator<1> > MXBUF4("Queue 4 selector",false);
CDevice<CNonRndNumGenerator<1> > MXBUF5("Queue 5 selector",false);
CDevice<CNonRndNumGenerator<2> > MXDEL1("Queue 1 selector delay",false);
CDevice<CNonRndNumGenerator<2> > MXDEL2("Queue 2 selector delay",false);
CDevice<CNonRndNumGenerator<2> > MXDEL3("Queue 3 selector delay",false);
CDevice<CNonRndNumGenerator<2> > MXDEL4("Queue 4 selector delay",false);
CDevice<CNonRndNumGenerator<2> > MXDEL5("Queue 5 selector delay",false);

// очереди
CLimitedQueue<CFIFODiscipline> BUF1("Queue 1", 10);
CLimitedQueue<CFIFODiscipline> BUF2("Queue 2", 10);
CLimitedQueue<CFIFODiscipline> BUF3("Queue 3", 10);
CLimitedQueue<CFIFODiscipline> BUF4("Queue 4", 10);
CLimitedQueue<CFIFODiscipline> BUF5("Queue 5", 10);
CLimitedQueue<CFIFODiscipline> TMPBUF("Selector unit temporary queue", 1, false);

// клапаны
CGate KBUF1("Queue 1 select gate");
CGate KBUF2("Queue 2 select gate");
CGate KBUF3("Queue 3 select gate");
CGate KBUF4("Queue 4 select gate");
CGate KBUF5("Queue 5 select gate");
CGate KLOST1("Lost messages gate 1");
CGate KLOST2("Lost messages gate 2");
CGate KLOST3("Lost messages gate 3");
CGate KLOST4("Lost messages gate 4");
CGate KLOST5("Lost messages gate 5");
CGate KMXON("Selector unit start gate",NANDCtrlFunc);
CGate KMXOFF("Selector unit stop gate");
CGate KSTART("Selector unit extra start gate");
CGate KSTOP("Selector unit extra stop gate",NANDCtrlFunc);

// функция регистрации Q-схемы и моделирования
...
{
    // регистрация элементов в Q-схеме
    REG_ELEM(MSG1);
    REG_ELEM(MSG2);
    REG_ELEM(MSG3);
    REG_ELEM(MSG4);
    REG_ELEM(MSG5);
    REG_ELEM(MXG);
    REG_ELEM(RG);
    REG_ELEM(BUF1);
    REG_ELEM(BUF2);
    REG_ELEM(BUF3);
    REG_ELEM(BUF4);
    REG_ELEM(BUF5);
    REG_ELEM(KBUF1);
    REG_ELEM(KBUF2);
    REG_ELEM(KBUF3);
    REG_ELEM(KBUF4);
    REG_ELEM(KBUF5);
    REG_ELEM(KLOST1);
    REG_ELEM(KLOST2);
    REG_ELEM(KLOST3);
    REG_ELEM(KLOST4);
    REG_ELEM(KLOST5);
    REG_ELEM(KMXON);
    REG_ELEM(KMXOFF);
```

```
REG_ELEM(MXBUF1);
REG_ELEM(MXBUF2);
REG_ELEM(MXBUF3);
REG_ELEM(MXBUF4);
REG_ELEM(MXBUF5);
REG_ELEM(MXDEL1);
REG_ELEM(MXDEL2);
REG_ELEM(MXDEL3);
REG_ELEM(MXDEL4);
REG_ELEM(MXDEL5);
REG_ELEM(KSTART);
REG_ELEM(KSTOP);
REG_ELEM(TMPBUF);

// регистрация связей

// информационные связи
REG_ILINK(MSG1,BUF1);
REG_ILINK(MSG1,KLOST1);
REG_ILINK(BUF1,KBUF1);
REG_ILINK(KBUF1,RG);
REG_ILINK(MSG2,BUF2);
REG_ILINK(MSG2,KLOST2);
REG_ILINK(BUF2,KBUF2);
REG_ILINK(KBUF2,RG);
REG_ILINK(MSG3,BUF3);
REG_ILINK(MSG3,KLOST3);
REG_ILINK(BUF3,KBUF3);
REG_ILINK(KBUF3,RG);
REG_ILINK(MSG4,BUF4);
REG_ILINK(MSG4,KLOST4);
REG_ILINK(BUF4,KBUF4);
REG_ILINK(KBUF4,RG);
REG_ILINK(MSG5,BUF5);
REG_ILINK(MSG5,KLOST5);
REG_ILINK(BUF5,KBUF5);
REG_ILINK(KBUF5,RG);
REG_ILINK(MXG,KMXON);
REG_ILINK(MXG,KMXOFF);
REG_ILINK(KMXON,TMPBUF);
REG_ILINK(TMPBUF,KSTOP);
REG_ILINK(TMPBUF,KSTART);
REG_ILINK(KSTART,MXBUF1);
REG_ILINK(MXBUF1,MXDEL1);
REG_ILINK(MXDEL1,MXBUF2);
REG_ILINK(MXBUF2,MXDEL2);
REG_ILINK(MXDEL2,MXBUF3);
REG_ILINK(MXBUF3,MXDEL3);
REG_ILINK(MXDEL3,MXBUF4);
REG_ILINK(MXBUF4,MXDEL4);
REG_ILINK(MXDEL4,MXBUF5);
REG_ILINK(MXBUF5,MXDEL5);

// управляющие связи
REG_CLINK(BUF1,KLOST1);
REG_CLINK(BUF2,KLOST2);
REG_CLINK(BUF3,KLOST3);
REG_CLINK(BUF4,KLOST4);
REG_CLINK(BUF5,KLOST5);
REG_CLINK_EX(MXBUF1,KBUF1,busy);
REG_CLINK_EX(MXBUF2,KBUF2,busy);
REG_CLINK_EX(MXBUF3,KBUF3,busy);
REG_CLINK_EX(MXBUF4,KBUF4,busy);
REG_CLINK_EX(MXBUF5,KBUF5,busy);
```

```
    REG_CLINK_EX(RG,KBUF1,empty);
    REG_CLINK_EX(RG,KBUF2,empty);
    REG_CLINK_EX(RG,KBUF3,empty);
    REG_CLINK_EX(RG,KBUF4,empty);
    REG_CLINK_EX(RG,KBUF5,empty);
    REG_CLINK_EX(BUF1,KMXON,empty);
    REG_CLINK_EX(BUF2,KMXON,empty);
    REG_CLINK_EX(BUF3,KMXON,empty);
    REG_CLINK_EX(BUF4,KMXON,empty);
    REG_CLINK_EX(BUF5,KMXON,empty);
    REG_CLINK_EX(BUF1,KMXOFF,empty);
    REG_CLINK_EX(BUF2,KMXOFF,empty);
    REG_CLINK_EX(BUF3,KMXOFF,empty);
    REG_CLINK_EX(BUF4,KMXOFF,empty);
    REG_CLINK_EX(BUF5,KMXOFF,empty);
    REG_CLINK_EX(MXBUF1,KSTART,empty);
    REG_CLINK_EX(MXBUF2,KSTART,empty);
    REG_CLINK_EX(MXBUF3,KSTART,empty);
    REG_CLINK_EX(MXBUF4,KSTART,empty);
    REG_CLINK_EX(MXBUF5,KSTART,empty);
    REG_CLINK_EX(MXDEL1,KSTART,empty);
    REG_CLINK_EX(MXDEL2,KSTART,empty);
    REG_CLINK_EX(MXDEL3,KSTART,empty);
    REG_CLINK_EX(MXDEL4,KSTART,empty);
    REG_CLINK_EX(MXDEL5,KSTART,empty);
    REG_CLINK_EX(MXBUF1,KSTOP,empty);
    REG_CLINK_EX(MXBUF2,KSTOP,empty);
    REG_CLINK_EX(MXBUF3,KSTOP,empty);
    REG_CLINK_EX(MXBUF4,KSTOP,empty);
    REG_CLINK_EX(MXBUF5,KSTOP,empty);
    REG_CLINK_EX(MXDEL1,KSTOP,empty);
    REG_CLINK_EX(MXDEL2,KSTOP,empty);
    REG_CLINK_EX(MXDEL3,KSTOP,empty);
    REG_CLINK_EX(MXDEL4,KSTOP,empty);
    REG_CLINK_EX(MXDEL5,KSTOP,empty);

    // задание параметров моделирования и запуск процесса моделирования
    const char * comment = "Communication unit model";
    SIMUL_STAT_FILE("results.stt", true, duration, 100000, comment);

...
}
//---------------------------------------------------------------------------
```

В приведенном проекте длины очередей и параметры генераторов, имитирующих потоки пакетов на входах модуля КС, взяты условно. Через NANDCtrlFunc обозначена функция объединения сигналов управления клапанами по И-НЕ. Заголовок функции регистрации Q-схемы и моделирования не показан, поскольку он зависит от вида моделирующего приложения. Например, в случае консольного приложения это будет функция main, а в случае приложения Windows – обработчик события кнопки пуска моделирования в графическом окне ПС.

На рисунке 7 показан внешний вид главного окна разработанной ПС с загруженной Q-схемой моделируемого модуля КС.

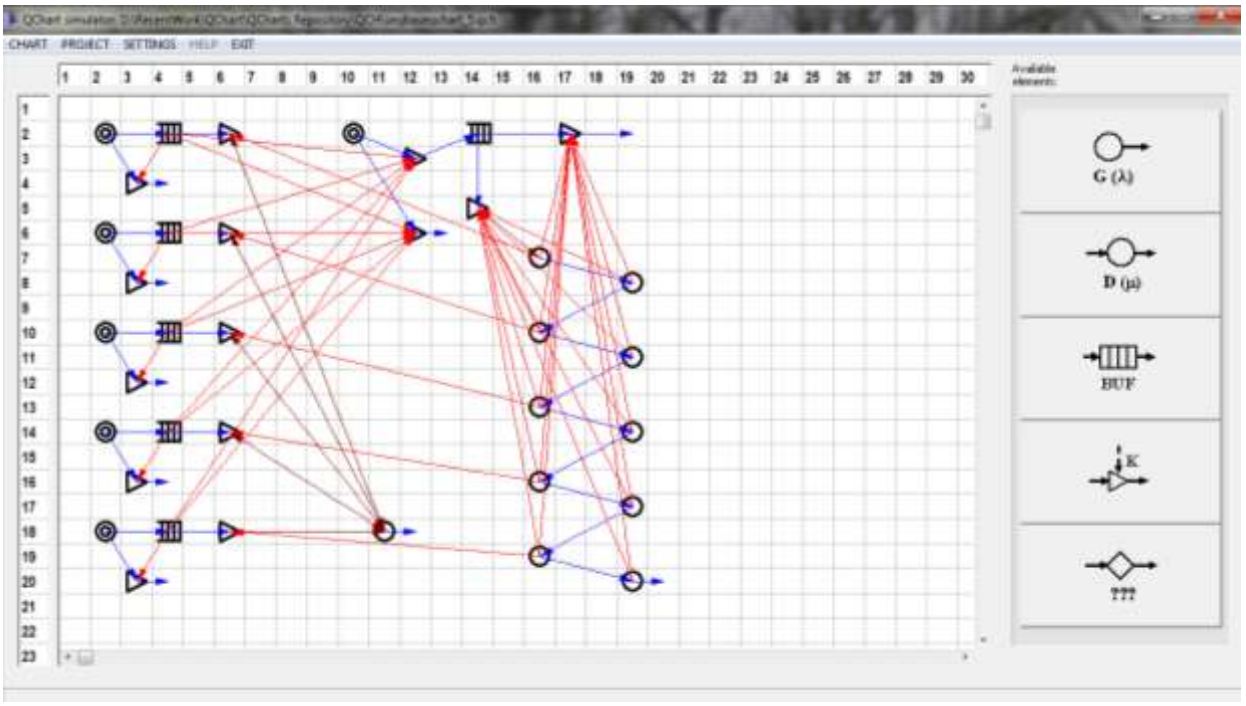

Рисунок 7. Внешний вид главного окна разработанной ПС с загруженной Q-схемой моделируемого модуля КС

В разработанной системе каждый элемент модели располагается в своей ячейке графической формы дизайнера Q-схем. Задать параметры элемента можно нажатием клавиши Enter или щелчком правой кнопки мыши на этом элементе. При этом появляется новое окно настройки. Его внешний вид зависит от типа элемента. Для генераторов заявок и каналов оно, к примеру, будет таким, как показано на рисунке 8.

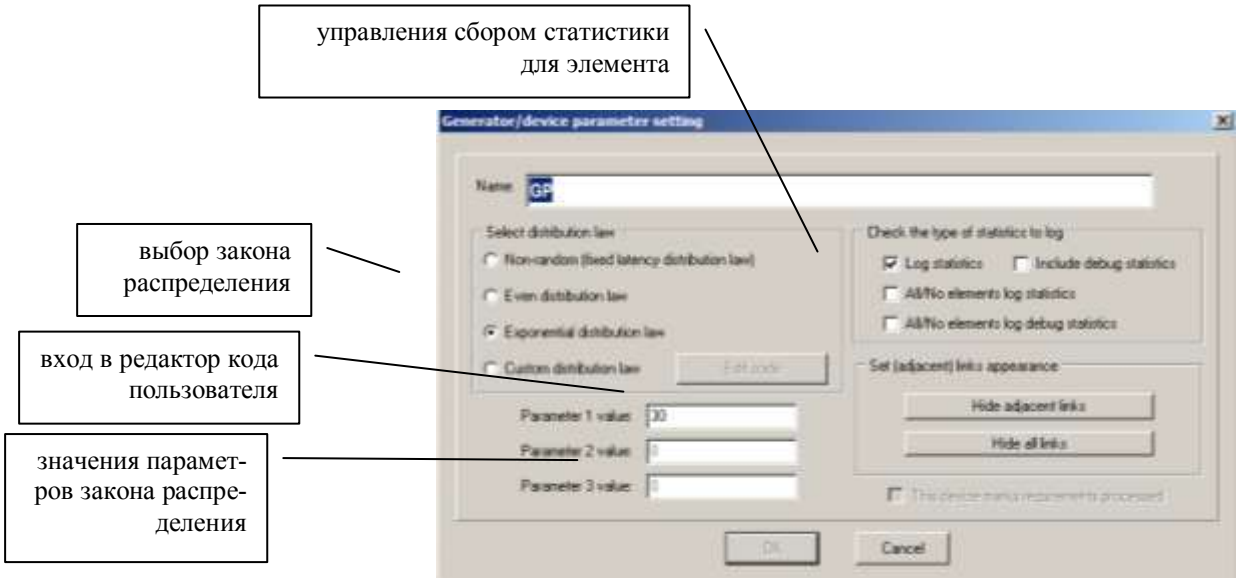

Рисунок 8. Окно настройки параметров генератора заявок / канала

Возвращаясь к главному окну ПС (рисунок 7), следует отметить, что все действия, относящиеся к работе с файлами Q-схем сведены в меню CHART, в меню PROJECT содержатся опции проектирования, проверки и настройки модели, в меню SETTING – опции настройки ПС и моделирования.

Таким образом, мы поэтапно рассмотрели процесс разработки модульной программной системы имитационного моделирования системы массового обслуживания с использованием объектной декомпозиции. Следует отметить, что разработанная ПС позволяет работать не только с данной конкретной Q-схемой, а с классом Q-схем, что позволяет проводить эксперименты с различными вариантами построения имитационной модели, исследуя те или иные характеристики системы.

5. Варианты заданий

Вариант **1**. Модель подсистемы ввода-вывода.

В подсистеме ввода-вывода имеется шесть накопителей на магнитных дисках (НМД) и два селекторных канала (СК). Запросы на обмен информацией с НМД распределены по закону Пуассона, обслуживание каждого запроса определяется начальной программой из трех команд. Первая команда обеспечивает подвод головок за время t_1, вторая — нахождение требуемой записи на МД за время t_2, третья — считывание или запись информации на МД за время t_3.

При поступлении запроса канал мгновенно передает первую команду нужному НМД и отсоединяется, а выбранный НМД будет в среднем за время t_1=15 мс выполнять первую команду. Если несколько НМД одновременно выполнили первую команду, то дальнейшее обслуживание осуществляется НМД с высшим приоритетом. После выполнения первой команды СК соединяется с нужным НМД и работает с ним вместе в течение времени t_2 и t_3, после чего канал отсоединяется от обслуживания НМД и готов к выполнению следующего запроса. Среднее значение t_2+t_3 равно 200 мс.

Выбор номера СК, который свяжется с нужным НМД, определяется наличием свободного в данный момент канала. Если свободными окажутся два канала сразу, то предпочтение отдается первому. Если оба канала заняты, то связь с НМД осуществляется через канал, освободившийся первым. Это касается обслуживания как

первой команды, так и двух оставшихся. Запросы на обмен поступают в среднем через 10 мс.

Смоделировать работу подсистемы ввода-вывода для N запросов. Определить загрузку каналов и НМД и среднее время обслуживания запроса.

Вариант **2**. Модель производственного цеха.

В производственном цехе работают три одинаковых станка. Во всех станках используется деталь, которая время от времени выходит из строя. Как только деталь отказывает, соответствующий станок необходимо выключить. Отказавшую деталь вынимают и на ее место ставят исправную запасную деталь сразу же или как только это станет возможным, и станок вновь включают. Неисправные детали ремонтируют и после ремонта используют снова.

Время работы детали распределено по нормальному закону со средним 350 часов и стандартным отклонением 70 часов. Съем отказавшей детали со станка занимает 4 часа. Время установки новой детали составляет 6 часов. Время ремонта неисправной детали распределено по нормальному закону со средним и стандартным отклонением соответственно 8 часов и 0.5 часов.

Ремонтом отказавших деталей занимается механик. В его обязанности входит также ремонт некоторых других деталей, поступающих из другого цеха. Эти другие детали поступают по закону Пуассона со средним интервалом между поступлениями 9 часов. Время их ремонта составляет 8±4 часа. Они имеют более высокий приоритет при ремонте, чем неисправные детали, используемые в рассматриваемых станках. Все три станка не зависят друг от друга за исключением процесса распределения деталей. Имеются только две запасные детали.

Смоделировать работу цеха в течение N лет, рассматривая 40-часовую рабочую неделю. Определить степень загрузки станков и механика. Найти среднюю часовую стоимость системы как сумму стоимости простоя станков и стоимости ожидания ремонта другими деталями. Считать, что стоимость простоя станка составляет 25 долларов в час, а стоимость ожидания ремонта другими деталями равна 5 долларам в час для каждой детали.

Вариант **3.** Модель локальной сети.

В локальной сети ЭВМ длина моноканала равна 2.5 км. Каждый абонент через свой адаптер может подключиться к моноканалу для передачи информационного кадра любому другому абоненту. Скорость распространения сигнала в моноканале 250 м/мкс. Скорость передачи сигналов в моноканал 10 Мбит/с. К сети подключено 6 абонентов с расстоянием между ними 500 м. Длина информационного кадра постоянна и равна 1000 бит. Нагрузка от каждого абонента 600 кадр/с и равномерно распределяется между отдельными абонентами.

Абоненты передают данные в канал по истечении некоторого времени после окончания передач других абонентов, время ожидания случайно и распределено равномерно на интервале 0–20 мкс. В случае одновременной передачи в канал кадров от двух или более абонентов происходит "столкновение" кадров и в кадрах возникают ошибки (искажения). Адаптер — приемник проверяет правильность каждого принятого кадра и посылает адаптеру — передатчику подтверждение того, что принятый кадр не искажен.

Кадр подтверждения имеет длину 100 бит и относительный приоритет при выдаче в моноканал перед информационными кадрами. Адаптер — передатчик ожидает подтверждения в течение времени передачи и распространения информационного кадра плюс 40 мкс. Ошибка в кадре подтверждения приводит к его потере. Следующий кадр от каждого абонента не может быть передан в моноканал до тех пор, пока не придет подтверждение на ранее переданный кадр или пока не истечет допустимое время ожидания получения подтверждения. Повторных попыток передачи неподтвержденных кадров не производится и такие кадры считаются утерянными.

Смоделировать передачу N кадров в сети. Определить долю потерь кадров и коэффициент использования моноканала, вычисляемый как отношение числа удачно переданных кадров к числу кадров, которые могли бы быть переданы за рассматриваемый интервал времени.

Вариант **4.** Модель справочно-информационной системы.

Автоматизированная справочно-информационная система содержит 3 терминала, процессор, основную память (ОП) и два накопителя на магнитных дисках (НМД). Имеется пять типов запросов, которые, как и ответы на них, поступают на терминалы 1, 2 и 3. Терминалы 1 и 2 обрабатывают сообщения, относящиеся к первым четырем типам запросов, а терминал 3 — сообщения, относящиеся к запросам пятого типа. Запросы типов 1, 2 и 5 характеризуются равномерным распределением интервалов между моментами поступления: 1 — 600±10 с; 2 — 720±20 с; 5 — 660±30 с, а запросы 3 и 4 являются пуассоновскими со средними значениями интервалов: 3 — 480 с; 4 — 540 с. Запросы через ОП и процессов записываются в НМД 1 за среднее время 20 с.

Существует 5 прикладных программ, хранящихся в НМД 2, каждая из которых обрабатывает сообщения, относящиеся к запросу определенного типа. При поступлении в ОП запроса с НМД 1 обрабатывающая этот запрос программа загружается в ОП, где одновременно могут находиться до 5 сообщений, прочитанных с терминалов, и до 8 сообщений, прочитанных с НМД 1. Процессор в каждый момент времени может обрабатывать только одно сообщение.

Процесс обработки запросов в системе включает следующие этапы: 1) формирование и передача запроса с терминала и запись его в НМД 1; 2) считывание запроса из НМД 1 в ОП; 3) загрузка в ОП с НМД 2 прикладной программы и поиск информации; 4) чтение найденной информации в ОП, ее обработка и вывод сообщения из ОП на терминал. Суммарное время обработки запроса в среднем составляет 60 с. Среднее время записи в НМД 1 — 20 с, записи с НМД 1 в ОП — 10 с; время поиска информации 30±5 с.

Смоделировать работу системы в течение N часов. Определить загрузки устройств, средние времена ожидания ответа на запрос, параметры очередей сообщений.

Вариант **5.** Модель городского гаража.

В городе имеется гараж, в котором производят работы по уходу и ремонту транспортных средств. Город небольшой, поэтому гараж оборудован только одной смотровой ямой и в нем работает

только один механик. Все городские транспортные средства обязаны регулярно пребывать в гараж для профилактического осмотра. Число автомобилей, пребывающих каждый день по графику осмотра, распределено равномерно от двух до четырех. Время обслуживания каждой машины равномерно распределено в интервале от 1.5 до 2.5 часов. Машины, направленные на осмотр в данный день, оставляют в гараже до начала каждого рабочего дня. Гараж открыт семь дней в неделю. Механик работает 8 часов в день без перерыва.

Город пытает поддерживать также полный парк полицейских машин. Требуется, чтобы полицейские машины можно было использовать 24 часа в сутки. Как только с какой-либо машиной что-то случается, ее немедленно доставляют в гараж для ремонта вне графика. Даже если на профилактическом осмотре в это время стоит другая машина, вновь прибывшей полицейской машине разрешается занять смотровую яму, и ее ремонт начинается без задержки. Тем не менее полицейская машина, нуждающаяся во внеплановом ремонте, не может вытеснить другую полицейскую машину.

Распределение времени между поступлением полицейских машин на внеплановый ремонт является пуассоновским со средним интервалом 48 часов. Если гараж закрыт в момент их поступления, машины должны ждать до 8 часов утра, пока не начнется работа. Время их обслуживания распределено экспоненциально со средним 2.5 часа.

Смоделировать работу гаража в течение N дней. Оценить распределение случайной переменной "число полицейских машин, находящихся на внеплановом ремонте".

Вариант **6.** Модель продовольственного магазина.

Продовольственный магазин состоит из трех прилавков и основной кассы при выходе. Кроме того, на выходе из магазина имеется экспресс-касса. Покупатели приходят в магазин в соответствии с распределением Пуассона. Среднее значение интервала прихода составляет 75 с. Войдя в магазин, каждый покупатель берет корзинку и может обойти один или несколько прилавков, отбирая продукты. Время, требуемое для обхода прилавка, и число покупок, выбранных у прилавка, распределены равномерно в соответствии с таблицей:

Прилавок	Вероятность покупки	Время обхода прилавка, с	Число покупок у прилавка, шт.
1	0.75	120±60	3±1
2	0.55	150±30	4±1
3	0.82	120±45	5±1

После того как товар отобран, покупатель становится в конец очереди к основной кассе. Стоя в очереди, покупатель может захотеть сделать еще 2±1 покупки. Покупатели с тремя и меньшим числом покупок (не считая возможной дополнительной покупки) пользуются экспресс-кассой. Все другие покупатели проходят через обычную кассу. Если у контролера экспресс-кассы нет работы, то покупателю из начала очереди у основной кассы разрешается перейти к экспресс-кассе. Время обслуживания покупателя в любой кассе пропорционально числу сделанных покупок, на одну покупку уходит 3 с проверки. После оплаты продуктов покупатель оставляет корзинку и уходит.

Построить модель, описывающую процесс покупок в продовольственном магазине. Провести моделирование N 8-часовых рабочих дней. Определить нагрузку кассиров и максимальные длины очередей перед кассой. Найти необходимое количество корзинок для бесперебойной работы магазина. Как изменятся указанные характеристики, если среднее значение интервала прихода покупателей составит T (T_0=30) с ?

Вариант **7**. Модель системы противовоздушной обороны.

Система противовоздушной обороны (ПВО) имеет в своем составе четыре радиолокационные станции наведения, каждая из которых обеспечивается двумя пусковыми установками. Каждая пусковая установка производит в среднем три выстрела в минуту. Средняя вероятность поражения цели одной выпущенной ракетой равна 0.57. Глубина зоны обстрела составляет 37 км и определяется как разность между средней дальностью перехвата и средней дальностью прекращения стрельбы. Глубина зоны обстрела одинакова для всех станций наведения.

Скорость налетающих самолетов при условии, что они не обстреливаются, — 850 км/ч. Средний линейный интервал между самолетами — 7 км. Одну цель могут обстреливать пусковые уста-

новки не более чем двух станций наведения. Каждая пусковая установка этих станций поражает цель независимо от других. За самолетом, влетевшим в зону обстрела, в случае занятости всех пусковых установок осуществляется слежение с помощью станций. Если какая-либо пусковая установка освобождается, то соответствующая станция передает самолет на обстрел при условии, что самолет еще находится в зоне обстрела.

Смоделировать работу ПВО при налете N самолетов. Определить вероятность поражения цели, среднее число ведущих обстрел пусковых установок. Считать, что налетающие самолеты образуют пуассоновский поток, а все заданные интервалы времени подчиняются экспоненциальному закону.

Вариант **8.** Модель заправочной станции.

На заправочную станцию прибывают автомашины со следующим распределением времени между прибытиями: меньше 0 — 0.0; 100 с — 0.25; 200 с — 0.48; 300 с — 0.69; 400 с — 0.81; 500 с — 0.90; 600 с — 1.0. Время обслуживания подчиняется следующему распределению: меньше 100 с — 0.0; 200 с — 0.06; 300 с — 0.21; 400 с — 0.48; 500 с — 0.77; 600 с — 0.83; 700 с — 1.0. (Через тире указана суммарная частота.) Машина останавливается для обслуживания лишь в том случае, если число ожидающих обслуживания автомашин меньше или равно числу обслуживаемых машин. Машины, которые не останавливаются, уезжают на другую заправочную станцию и, таким образом, уменьшают возможную прибыль.

Заправочная станция открыта с 7.00 до 19.00. Машины, поступившие позднее 19.00 не обслуживаются. Тем не менее все машины, попавшие в очередь до 19.00, должны быть обслужены. Прибыль с одной обслуженной машины составляет в среднем 1 доллар, включая заработок служащих и другие постоянные расходы. На каждой колонке работает один служащий. Заработок служащих составляет 2.5 доллара в час и выплачивается только за 12-часовой рабочий день, даже если они задерживаются после 19.00 для того, чтобы закончить обслуживание ждущих машин. Другие постоянные расходы составляют 75 долларов в день.

Смоделировать работу заправочной станции в течение N дней. Определить, сколько служащих следует нанять владельцу станции, чтобы получить максимальную дневную прибыль.

Вариант **9**. Модель производства продукта.

Некоторый производимый продукт получается в результате соединения двух, пяти и трех частей типов A, B и C соответственно. Время между поступлениями этих частей в место соединения равно соответственно: A — 15±5 мин, B — 6±2 мин, C — 10±3 мин.

Операция соединения включает следующие этапы:

Объединить одну часть A и две части B. В результате получается полуфабрикат D. Требуемое время 15±3 мин.

Объединить по одной части A, B и C. Получается полуфабрикат E. Требуемое время 18±3 мин.

Проверить D и E на соответствие. Требуемое время 5±2 мин.

Объединить полуфабрикат D с одной частью B. Получается полуфабрикат F. Требуемое время 10±2 мин (этап 4 не может начаться, пока не завершен этап 3).

Объединить полуфабрикаты F и E. Результат — G. Требуемое время 15±4 мин.

Объединить G с одной частью B и двумя частями C. Соединение завершено. Требуемое время 8±3 мин.

Каждый из перечисленных этапов может быть выполнен одним рабочим. Один и тот же рабочий может переключаться с этапа на этап.

Промоделировать производство продукта при поступлении N частей и определить минимальное число рабочих, необходимое для того, чтобы избежать излишнего накопления частей A, B и (или) C в месте их соединения.

Вариант **10.** Модель переговорного пункта.

Междугородний переговорный пункт в небольшом городке имеет три телефонных аппарата для переговоров. Переговоры бывают двух видов: обычные и срочные. До начала переговоров клиент оформляет заказ на разговор в течение 2±1 мин. Относительный приоритет имеют оформляющие срочный разговор. Один из телефонных аппаратов используется только для срочных переговоров. Если заказан срочный разговор, а указанный аппарат занят, клиент вне очереди занимает первый освободившийся аппарат. Клиенты,

заказавшие обычные переговоры, становятся в кратчайшую очередь.

В среднем за сутки поступает 200 заявок на обычные переговоры и 60 — на срочные переговоры. Средняя длительность переговоров обоих видов (с учетом вызова абонента в другом городе) составляет 7 мин. В 10% случаев вне зависимости от вида разговора вызываемый абонент отсутствует на месте. Средняя длительность вызова отсутствующего абонента равна 1 мин. По истечении этого времени клиент, заказавший разговор, возвращается к столу заказов для переоформления заказа в течение 1 мин, а затем покидает переговорный пункт. Переоформление заказа на другое время имеет высший приоритет.

Смоделировать работу переговорного пункта в течение N суток. Определить среднее число ожидающих срочного и обычного разговоров, соответствующие средние времена ожидания, среднее число людей на переговорном пункте, среднее число людей в очереди к столу заказов. Считать все интервалы времени распределенными по экспоненциальному закону.

Вариант **11.** Модель библиотеки.

В библиотеке без открытого доступа лица желающие получить книги приходят к столу по закону Пуассона со средней интенсивностью 30 человек в час. Каждый хочет получить ровно одну книгу. Требуемая книга всегда имеется. У стола выдачи работают четыре библиотекаря. Как только библиотекарь освобождается, он может взять листки запроса не более чем у четырех человек одновременно (если такое число людей ожидает обслуживания). Время передачи листков запроса несущественно. Время прохождения от стола выдачи до хранилища (в один конец) равно 1±0.5 мин. Время поиска одной, двух, трех или четырех книг распределено по нормальному закону со средним соответственно 3, 6, 9 и 12 мин и стандартным отклонением, равным 20% от среднего.

После возвращения библиотекаря из хранилища время, требуемое на завершение процедуры выдачи, составляет 2±1 мин на человека. Листки запроса берут в порядке поступления. Завершение процедуры выдачи книг происходит аналогично приему заказов. Если свободны два или более библиотекарей и прибывает посетитель, то его обслуживает тот, кто был свободен дольше других. Ес-

ли свободны два или более библиотекарей и обслуживания ожидают два или более посетителей, то библиотекари не делят работу поровну. Вместо этого один библиотекарь берет столько листков, сколько сможет (но не более четырех); затем, если остаются еще посетители с листками, то следующий библиотекарь также берет столько листков, сколько сможет, и т.д.

Смоделировать процесс обслуживания N человек. Определить распределение случайных величин "время, проведенное каждым читателем у стола выдачи" и "число листков, забираемых библиотекарем". Найти нагрузку библиотекарей.

Вариант **12.** Модель работы станка.

В станке используют две детали A и B, которые периодически выходят из строя. Как только деталь A или B отказывает, станок отключают. Затем отказавшую деталь вынимают, вместо нее ставят исправную запасную (если она имеется или как только она появится) и станок вновь включают. И деталь A, и деталь B можно отремонтировать и использовать снова. Время работы деталей A и B распределено по нормальному закону со средним и стандартным отклонением соответственно: A — 350 час и 70 час; B — 450 час и 90 час. Съем любой отказавшей детали со станка занимает 4 час, а установка заменяющей ее детали этого же типа требует 6 час. Время ремонта неисправной детали A распределено по нормальному закону со средним 8 час и стандартным отклонением 0.5 час, а время ремонта детали B в соответствии с таблицей:

Время ремонта, час	**менее 5**	**6**	**7**	**8**	**9**
Суммарная частота	0.00	0.22	0.57	0.83	1.00

Ремонт деталей A и B в порядке их поступления выполняет механик. Кроме того, он ремонтирует другие детали, имеющие при ремонте высший приоритет. Эти другие детали поступают по закону Пуассона со средним интервалом 9 час. Время, требуемое на их ремонт, составляет 8±4 час. Имеется две запасные детали типа A и одна запасная деталь типа B.

Смоделировать работу станка в течение N лет, считая, что рабочая неделя состоит из 40 час. Определить загрузку станка, загрузку механика, среднюю длину очереди деталей к механику.

Вариант **13**. Модель коммерческого банка.

Приход клиентов в коммерческий банк описывается пуассоновским входящим потоком со средней интенсивностью 200 в час. В течение всего времени работы открыто шесть окошек кассиров. К каждому кассиру стоит индивидуальная очередь. Если в момент входа клиента в банк хотя бы один кассир свободен, клиент сразу же попадает к этому кассиру. В противном случае клиент присоединяется к любой очереди, которая на текущий момент является кратчайшей. Обслуживание клиентов осуществляется в порядке их поступления. Если клиент, уже стоящий в конце своей очереди, видит другую очередь, в которой стоит меньше человек, то он уходит из своей очереди и встает в конец самой короткой. После обслуживания клиент уходит из банка. В процессе перераспределения очередей высший приоритет имеют клиенты, уже находящиеся в банке.

Обслуживание в кассе может быть разделено на пять видов:

Вид операции	Относительная частота	Среднее время обслуживания, с
1	0.10	45
2	0.19	75
3	0.32	100
4	0.24	150
5	0.15	300

Время обслуживания каждого вида имеет экспоненциальное распределение; ни один из клиентов не требует выполнения более чем одного вида обслуживания за один визит в банк.

Смоделировать работу банка в течение Z отдельных пятичасовых рабочих дней. Определить загрузку кассиров, средние длины очередей.

Вариант **14.** Модель магистрали передачи данных.

Магистраль передачи данных состоит из трех каналов (основного и двух резервных) и общего накопителя. При нормальной работе магистрали сообщения передаются по основному каналу за 8±4 с. В основном канале происходят сбои через интервалы времени 150±30 с. Если сбой происходит во время передачи, то в течение 3 с запускается первый резервный канал, который передает прерванное сообщение с самого начала. В нем также возможны сбои через интервалы времени 170±40 с. При сбое во время передачи за 5 с запускается абсолютно надежный второй резервный канал. Восстановление основного канала занимает 30±10 с, а первого резервного — 20±5 с.

После восстановления второй резервный канал, передав сообщение, выключается и работу с очередным сообщением начинает основной канал (если он восстановлен). Аналогично, первый резервный канал, передав сообщение (при отсутствии в нем сбоев), выключается и основной канал продолжает работу с очередного сообщения. Сообщения поступают через 10±5 с и остаются в накопителе до окончания передачи. В случае сбоя в первом резервном канале второй резервный канал также передает прерванное сообщение с самого начала.

Смоделировать работу магистрали передачи данных в течение R часов. Определить загрузку запасных (резервных) каналов, число прерванных сообщений. Найти объем накопителя, гарантирующий передачу сообщений с вероятностью потери меньше 0.1.

Вариант **15**. Модель аэропорта.

Аэропорт обслуживает 120 самолетов. Поток посадок самолетов и поток поступления самолетов по рейсовым регламентам из числа совершивших посадку является простейшим и стационарным. Плотность поступления самолетов на техническое обслуживание равна 2 самолета в сутки. Обслуживание ведется круглосуточно двумя специализированными бригадами. Каждый самолет вначале осматривается первой бригадой, производящей регламентные профилактические работы двигателей, с производительностью 3 самолета в сутки. Вторая бригада производит регламентные рабо-

ты по планеру, шасси и др. с производительностью 4 самолета в сутки.

Промоделировать обслуживание в аэропорту самолетов. Определить вероятность того, что обе бригады будут свободны от обслуживания, среднее число самолетов, находящихся в системе обслуживания, среднее число самолетов, обслуживаемых обеими бригадами в сутки, среднее число самолетов, ожидающих обслуживания, время простоя первой и второй бригад в течение суток.

Вариант **16**. Модель рыболовецкого предприятия.

Рыболовецкое предприятие вылавливает в среднем около 10 тонн рыбы в сутки. Улов зависит от ряда случайных факторов, поэтому можно считать, что поток выловленной рыбы имеет пуассоновское распределение. Часть рыбы предприятие перерабатывает на собственном пункте, состоящем из двух одинаковых цехов. Производительность каждого цеха 2 тонны в сутки. Остальную часть улова предприятие отгружает для отправки на завод по переработке рыбы. На предприятии имеется холодильник для хранения рыбы объемом 2 тонны, где она может храниться не более суток. Если за это время рыбу не начали обрабатывать, то ее немедленно отправляют на другой завод, который производит ее копчение. В противном случае она испортится.

Промоделировать работу предприятия в течение N недель. Определить, сколько в среднем надо выделить автомашин для перевозки рыбы, если при соответствующем состоянии дорог и расстоянии до завода переработки машина грузоподъемностью 2.5 тонны может сделать 2 рейса в сутки, а до завода копчения — один рейс в сутки.

Вариант **17**. Модель боевого столкновения.

В боевом столкновении у нападающей стороны два образца вооружения. Для обстрела маневрирующей цели в среднем требуется 2 минуты. Вероятность поражения цели при обстреле равна 0.9. Нападающая сторона обладает системой разведки, позволяющей в среднем обнаружить одну цель в минуту. Среднее время пребывания цели в зоне обстрела после ее обнаружения равно 4 минутам. Независимо от того, обстреливается ли цель или нет, она по

истечении определенного времени покидает свою позицию и для нападающей стороны считается потерянной, так как поражена быть не может.

Промоделировать Z часов боевого столкновения. Определить эффективность вооружения нападающей стороны по поражению появляющихся целей. Определить также, сколько единиц вооружения необходимо иметь нападающей стороне, чтобы ее потери не превышали в среднем 10%, если цель противника в случае непоражения может нанести ответный удар с эффективностью 0.5.

Вариант **18**. Модель воздушного боя.

Через систему ПВО прорывается группа самолетов, состоящая из четырех постановщиков помех и одного бомбардировщика. Огонь сначала ведется по постановщикам помех, а затем по бомбардировщику. Поток пусков ракет можно считать пуассоновским с интенсивностью 0.5. Вероятность поражения одной ракетой одного самолета (любого) равна 0.7. После поражения самолета огонь мгновенно переносится на следующий самолет. Время нахождения самолетов в зоне обстрелов ПВО равно 5 минутам.

Промоделировать воздушный бой. Определить вероятность того, что за время обстрела бомбардировщик будет поражен. Проанализировать как изменится эта вероятность, если вероятность поражения постановщиков помех увеличится до 0.8, а вероятность поражения бомбардировщика снизится до 0.6.

Вариант **19.** Модель дозаправки самолетов.

Производится дозаправка самолетов горючим в воздухе. В районе дозаправки находятся 3 самолета-заправщика. Самолет, нуждающийся в дозаправке, входит в район, где дежурят заправщики, с вероятностью 0.9. Если самолет вышел в район дозаправки т свободен хотя бы один из заправщиков, то он производит дозаправку, на что уходит в среднем 10 мин. Если самолет выходит в район дозаправки, но все самолеты-заправщики заняты, то этот самолет остается недозаправленным. Во всех случаях, когда самолет недозаправлен, он вынужден садиться на запасной аэродром. Самолеты подходят к району дозаправки со средним интервалом в 150 секунд.

Смоделировать процесс дозаправки самолетов в течение N часов. Определить вероятность того, что самолет будет вынужден садиться на запасной аэродром.

Вариант **20**. Модель ремонтной станции.

На станцию текущего ремонта сельскохозяйственной техники поступают в случайные моменты времени различные машины. Станция имеет одно помещение для одного технологического потока машин. Во дворе станции имеется небольшая крытая площадка, где одновременно может находиться, ожидая очереди, не более двух машин. В зависимости от характера неисправности, наличия запасных частей и квалификации ремонтных рабочих время на ремонт каждой машины затрачивается случайное. Статистика ремонтного времени в этой мастерской показала, что оно распределено по показательному закону со средним значением 1.5 суток. Неисправная техника поступает в мастерскую из близлежащих хозяйств. Случайные, независимые друг от друга, моменты выхода из строя сельскохозяйственных машин и разное удаление мест их эксплуатации от ремонтной мастерской позволяют предположить, что поток поступающей неисправной техники в мастерскую простейший со средней плотностью 0.5 машин в сутки.

Промоделировать работу мастерской в течение нескольких недель. Определить пропускную способность мастерской, среднее время простоя мастерской, среднее число машин, ожидающих ремонта. Проанализировать, насколько изменятся эти характеристики, если оборудовать второе помещение с новым технологическим потоком для ремонта техники.

Вариант **21**. Модель вычислительного центра.

В вычислительный центр (ВЦ) поступают заказы на проведение метеорологических расчетов. Так как прогноз погоды имеет смысл производить по свежим метеоданным, поступающая информация со временем стареет. По этой причине получаемая информация сохраняет определенную достоверность в течение 10 часов. Плотность поступления заказов на проведение расчетов составляет 2 заказа в час. ВЦ для проведения этих работ может использовать две ЭВМ. Время решения на первой ЭВМ одной задачи в среднем

равно 5 часам, а на второй ЭВМ — 1 час. Это время имеет показательный закон распределения.

Промоделировать функционирование ВЦ в течение F часов. Определить вероятность того, что две машины свободны от проведения расчетов, вероятность того, что одна из машин будет занята расчетом, а другая свободна, вероятность того, что поступившие данные для расчетов не будут сразу же использованы.

Вариант **22**. Модель производственного процесса.

Процесс деталей определенного вида включает длительный процесс сборки, заканчивающийся коротким периодом обжига в печи. Так как содержание печи обходится довольно дорого (80 долларов за 8-часовой рабочий день независимо от степени использования), то несколько сборщиков используют одну печь, в которой одновременно можно обжигать только одну деталь. Сборщик не может начать новую сборку пока не вытащит из печи предыдущую деталь. Операция сборки занимает в среднем 30 минут, а операция обжига — 8 минут. Зарплата сборщика составляет 20 долларов в час, цена материала — 5 долларов за деталь, стоимость готового изделия — 15 долларов.

Промоделировать производственный процесс в течение N рабочих дней. Определить оптимальное число сборщиков, использующих одну печь (т.е. такое число, при котором производство дает максимальную прибыль).

Вариант **23**. Модель заправочной станции.

На заправочную станцию прибывают автомашины со средним интервалом между прибытиями 200 секунд. Среднее время обслуживания равно 400 секунд. Машина останавливается на обслуживание только в том случае, если число ожидающих обслуживания автомашин меньше или равно числу обслуживаемых машин. Машины, которые не останавливаются, уезжают на другую заправочную станцию и, таким образом, уменьшают прибыль. Автозаправочная станция работает 12 часов в день. Прибыль с одной обслуженной машины составляет в среднем 5 долларов, включая заработок служащих и другие постоянные расходы. На каждой колонке

работает один служащий с заработком 10 долларов в час. Другие постоянные расходы составляют 125 долларов в день.

Промоделировать работу автозаправочной станции в течение дня. Определить, сколько служащих следует нанять владельцу станции, чтобы получить максимальную дневную прибыль.

Вариант **24.** Модель автохозяйства.

Автохозяйство имеет пункт автоматической мойки машин, однако он полностью не обеспечивает мойку всех машин автохозяйства. Поэтому, чтобы не загромождать подъездные пути к базе, решено организовать работу следующим образом: машина, прибывая после работы в парк, поступает на пункт автомойки, если там есть место. В противном случае машина становится на свое место на стоянке, где и моется. Машина, которая была вымыта на пункте автомойки, отправляется на пункт заправки, если он свободен. В противном случае машина идет к месту стоянки, а заправка ее производится перед выходом в рейс на следующий день. Поток прибывающих машин в парк равен 4 машины в минуту. Пункт автомойки имеет 4 рабочих места одинаковой производительности 0.5 машины в минуту. Пункт заправки имеет одно место производительностью 2 машины в минуту.

Смоделировать работу автохозяйства в течение нескольких дней. Определить процент машин, которые проходят мойку на пункте автомойки и заправляются горючим до постановки их на стоянку.

Вариант **25**. Модель газетного киоска.

В газетный киоск газеты поставляют только в кипах по 25 штук. Каждая газета стоит 10 центов. Киоск платит 7 центов за каждую проданную газету и 4 цента за каждую непроданную (непроданные газеты имеют определенную ценность, так как их можно вернуть на бумажную фабрику). Считается, что запрос на газеты возникает из двух источников. Первый источник — это постоянные покупатели; их потребность в газетах в среднем 175 штук в день. Другой источник — случайные покупатели, чья потребность в газетах определяется наличием особых событий, происшедших за день. Запрос от случайных покупателей распределен как показано ниже:

	Особые события дня		
	нет	тип 1	тип 2
Частота возникновения	0.3	0.5	0.2
Среднее число запросов	0	50	100

Предполагается, что вышеизложенная информация — это все, что учитывает владелец киоска.

Промоделировать работу газетного киоска в течение нескольких дней. Определить, сколько газетных кип следует брать владельцу ежедневно, если он хочет иметь максимальный доход.

Вариант **26**. Модель морского боя.

Соединение надводных кораблей, имеющих средства противоракетной обороны в количестве 6 единиц, подверглось нападению катеров противника. Управление отражением огня организовано так, что обстрел катеров ведется сосредоточенным огнем всех средств по одному катеру. Плотность поступления катеров в зону обстрела кораблей составляет 20 катеров в минуту. Время, необходимое одной единице вооружения на обстрел катера, равно 0.1 минуты. Каждое средство поражает катер за одну стрельбу с вероятностью 0.3.

Смоделировать морской бой в течение нескольких десятков минут. Определить процент обстрелянных катеров противника, а также среднюю вероятность поражения каждой цели.

Вариант **27**. Модель радиолокационного комплекса.

Радиолокационный комплекс измерения параметров движения самолетов по трассе состоит из двух станций, каждая из которых может измерять параметры движения самолетов в пределах зоны протяженностью 50 км. Скорость самолетов по трассе 1000 км/час. Параметры самолета измеряются в среднем в течение одной минуты. Если самолет, влетевший в зону измерения параметров, застанет обе РЛС занятыми, то его параметры не измеряются. В среднем за один час в зону измерения параметров влетает 30 самолетов.

Смоделировать работу радиолокационного комплекса в течение N часов. Определить вероятность того, что параметры самолета

будут измерены, среднее число занятых РЛС, вероятность того, что РЛС занята, среднее время простоя РЛС.

Вариант **28**. Модель овощной базы.

На городскую овощную базу поступают овощи нового урожая. Для обеспечения длительного хранения они проходят стадию обработки. До обработки они хранятся под открытым небом. После транспортировки овощи могут храниться без существенной потери качества не больше суток. Поток поступления овощей на базу можно считать пуассоновским, в среднем на базу в течение декады прибывает до 25 автомашин овощей двухтонной грузоподъемности. На базе работает две бригады по предварительной обработке овощей перед закладкой их на длительное хранение. Каждая бригада способна переработать за сутки до 3 тонн овощей.

Смоделировать работу базы в течение F суток. Найти вероятность того, что обе бригады будут без дела из-за отсутствия на базе овощей, вероятность того, что привезенные овощи не будут своевременно обработаны, среднее число бригад, занятых обработкой овощей, коэффициенты загрузки и простоя бригад.

Вариант **29.** Модель контрольно-проверочного комплекса.

Для выборочной проверки качества выпускаемых изделий на предприятии имеется автоматизированный контрольно-проверочный комплекс (КПК). Он устроен так, что в нем одновременно могут находиться не более трех изделий. Одно из них находится на проверке соответствия основных параметров изделия заданным, а два других ожидают проверки. Если очередное выпускаемое изделие застанет в КПК три изделия, то оно поступает на склад без проверки. Время проверки одного изделия в среднем 0.5 часа. Предприятие выпускает в среднем 2 изделия в час. КПК время от времени выходит из строя. Среднее время его безотказной работы равно 10 часам. При выходе из строя КПК изделие, которое находилось на проверке, поступает на склад полностью непроверенным. Среднее время восстановления работоспособности КПК равно одному часу.

Промоделировать работу КПК в течение нескольких рабочих дней. Найти средний процент проверенных изделий, среднее число

изделий, находящихся в КПК, а также средний процент изделий, которые были не проверены в силу отказа КПК.

Вариант **30.** Модель пункта первичной санитарной обработки.

Для проведения первичной санитарной обработки личного состава, подвергшегося заражению, создается полевой пункт санитарной обработки. Время на обработку одного человека на пункте составляет 0.1 часа. Поток поступления личного состава, требующего обработки, является простейшим с интенсивностью 100 человек в час. Пункт оборудуется укрытием для личного состава на 5 человек. Половина личного состава, т.е. около 50 человек в час, отправляются медицинским транспортом в тыл на стационарный пункт санитарной обработки. Кроме того, около 10% личного состава может быть отправлено в тыл попутным транспортом. Учитывая, что состояние организма человека зависит от времени, прошедшего с момента заражения, время ожидания начала обработки не должно превышать 10 часов.

Промоделировать работу пункта первичной санитарной обработки, варьируя его пропускную способность. Определить необходимую пропускную способность пункта.

6. Контрольные вопросы

1. Какое из сформулированных ниже утверждений верно?

a - отладка программы, написанной с применением функциональной декомпозиции, значительно проще, чем отладка объектной программы
b - возможности повторного использования кода при использовании объектной технологии программирования значительно выше, чем в случае использования функциональной декомпозиции
c - объем программного кода при использовании объектной технологии программирования существенно меньше, чем в случае использования функциональной декомпозиции
d - функциональная декомпозиция обеспечивает более высокую надежность программ, чем объектная технология
e - программа, написанная по объектной технологии, не может создаваться коллективом разработчиков параллельно во времени

2. Жизненный цикл программного обеспечения представляет собой:

a - период времени от момента зарождения идеи создания ПО до момента его внедрения в эксплуатацию
b - период времени от момента подписания технического задания на создание ПО до момента завершения его поддержки
c - период времени от момента зарождения идеи создания ПО до момента завершения его поддержки
d - период времени от момента подписания технического задания на создание ПО до момента оформления эксплуатационной документации
e - период времени от момента подписания технического задания на создание ПО до момента завершения его эксплуатации

3. Какие из перечисленных ниже факторов определяют надежность программы?

a - число не выявленных ошибок проектирования и используемая технология разработки
b - только используемая технология разработки
c - только число не выявленных ошибок проектирования
d - длина идентификаторов
e - число используемых препроцессорных директив

4. Стиль программирования включает:

a - только правила распределения и заполнения комментариев
b - только правила выбора идентификаторов
c - правила выбора идентификаторов и правила распределения и заполнения комментариев
d - правила выбора систем счисления при инициализации констант
e - правила выбора типов данных

5. Каким образом качество программной системы зависит от используемой технологии разработки?

a - скачкообразно растет при переходе к более совершенной техноло-

гии разработки
b - зависимости нет
c - линейно возрастает при переходе к более совершенной технологии разработки
d - полиномиально возрастает при переходе к более совершенной технологии разработки
e - зависимость носит случайный характер

6. Проектирование программной системы имеет целью:

a - разработку внутренней структуры программной системы и основных алгоритмов
b - разработку внутренней структуры программной системы и основным форматов данных
c - разработку внутренней структуры программной системы без детализации компонент
d - разработку внутренней структуры программной системы и интерфейса пользователя
e - разработку внутренней структуры программной системы с детализацией компонент

7. Какие из перечисленных ниже факторов определяют временную эффективность программы: 1) временная сложность алгоритма решения задачи; 2) количество файлов в проекте программы; 3) правила выбора идентификаторов для программных сущностей; 4) количество и число итераций циклов программы; 5) число глобальных переменных в программе?

a - только четвертый
b - только пятый
c - первый и четвертый
d - первый и пятый
e - только первый

8. Какое из сформулированных ниже утверждений верно?

a - тестирование является доказательством правильности программы
b - тестирование не требует наличия эталона

c - зависимость времени полного тестирования программы от объема входных данных линейная
d - тестирование является одной из фаз отладки программы
e - зависимость времени полного тестирования программы от объема входных данных линейно-логарифмическая

9. Тест программы представляет собой:

a - заранее подготовленный набор входных значений и соответствующий ему результат работы программы
b - заранее подготовленный набор входных значений и описание симптомов искомой ошибки
c - представительную выборку входных значений и соответствующие ее результаты работы программы
d - заранее подготовленный набор входных и промежуточных значений
e - заранее подготовленный набор алгоритмов проверки программы

10. Какое из перечисленных ниже требований и условий наиболее существенно для выполнения отладки программы?

отладка программы может выполняться без использования средств автоматизации
до перехода к отладке необходимо устранить все технологические ошибки в тексте программы
до перехода к отладке необходимо исправить все ошибки разработанной программной документации
до перехода к отладке необходимо во всех деталях понять задачу, решаемую программой
отладка программы может выполняться без непосредственного ее выполнения на компьютере

7. Содержание отчета

Результаты выполнения задания представляются в виде единого отчета о лабораторном практикуме, который должен включать следующие документы:

- задание к лабораторному практикуму;
- имитационная модель системы массового обслуживания;
- результаты объектной декомпозиции проектируемой системы (иерархия классов и объектная структура);
- результаты функциональной декомпозиции модели (проект модульной структуры программной системы);
- сценарий диалога с пользователем и дизайн меню программной системы;
- содержание основных классов и программных модулей, наиболее важные алгоритмы;
- описание методики тестирования ПС и результаты тестов (таблицы тестирования, снимки экрана и т.д.);
- листинг основных модулей программной системы на языке C++.

Все документы оформляются в текстовом редакторе (например, MS Office Word или LibreOffice Writer) на листах формата A4, распечатываются на одной стороне листа, и представляются в виде жестко скрепленного отчета по лабораторному практикуму.

Документы по содержанию должны соответствовать ГОСТ 34.201-89, 34.602-89, 19.701-90.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Иванова, Галина Сергеевна. Технология программирования [Текст] : учебник / Г. С. Иванова. - М. : Кнорус, 2011. - 336 с.

2. Технология программирования [Электронный ресурс] : учебное пособие / Ю. Ю. Громов, О. Г. Иванова, М. П. Беляев, Ю. В. Минин. – Тамбов : Изд-во ТГТУ, 2013. - 173 с. - Режим доступа: http://biblioclub.ru/index.php?page=book_red&id=277802

3. Липаев В. В. Проектирование и производство сложных заказных программных продуктов [Текст]. - М. : СИНТЕГ, 2011. – 408 с.

4. Липаев, В. В. Процессы и стандарты жизненного цикла сложных программных средств [Текст] : справочник / В. В. Липаев. - М. : СИНТЕГ, 2006 (Люберцы). - 260 с.

5. Липаев, В. В. Документирование сложных программных средств [Текст] : монография / В. В. Липаев. - М. : СИНТЕГ, 2005. - 200 с.

6. Запорожец, Сергей Александрович. Основы разработки и стандартизации программных средств. Пакеты прикладных программ в экономике [Текст] : учебное пособие / С. А. Запорожец, А. Т. Миргалеев ; ЮЗГУ. - Курск : ЮЗГУ, 2010. - 217 с.