МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение
высшего образования
«Курганский государственный университет»

Кафедра программного обеспечения автоматизированных систем

МЕТОДЫ ВЕРИФИКАЦИИ И ОЦЕНКИ КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Методические указания
к выполнению практических работ
для магистрантов направления подготовки 09.04.04
«Программная инженерия»

Курган 2020

Кафедра: «Программное обеспечение автоматизированных систем»

Дисциплина: «Методы верификации и оценки качества программного обеспечения», направление 09.04.04 «Программная инженерия», направленность «Методы и алгоритмы интеллектуальной обработки данных в информационно-вычислительных системах»

Составил: канд. техн. наук, доцент А. М. Семахин.

Утверждены на заседании кафедры «21» июня 2019 г.

Рекомендованы методическим советом университета «14» марта 2019 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Дисциплина «Методы верификации и оценки качества программного обеспечения» имеет целью формирование у студентов теоретических знаний и практических навыков в применении методов подтверждения соответствия программного продукта заявленным требованиям и оценки показателей качества программы.

Предмет дисциплины – технология оценки качества программных приложений.

Задачи дисциплины:

- изучение методов подтверждения соответствия программ требованиям спецификации;
- изучение методов, технологий и средств разработки и контроля качества процесса и результатов проектирования программ;
- изучение методов и средств измерения и оценки показателей качества программ, предъявленных для эксплуатации пользователям.

Практические занятия (36 часов).

Методические указания содержат теоретическое обоснование и варианты заданий для выполнения практических работ по дисциплине «Методы верификации и оценки качества программного обеспечения».

Методические указания разработаны в соответствии с требованиями государственного образовательного стандарта по подготовке магистрантов по направлению 09.04.04 «Программная инженерия».

1 Основные понятия и определения

Верификация (verification process) – процесс определения соответствия программных продуктов требованиям.

В процессе верификации проверяются условия:

- непротиворечивость требований к системе и степень учёта потребностей пользователей;
- возможность поставщика выполнить заданные требования;
- соответствие выбранных процессов жизненного цикла программных средств условиям договора;
- адекватность стандартов, процедур и среды разработки процессам жизненного цикла программных средств;
- соответствие проектных спецификаций программных средств заданным требованиям;
- корректность описания в проектных спецификациях входных и выходных данных, последовательности событий, интерфейсов и логики;
- соответствие кода проектным спецификациям и требованиям;
- тестируемость и корректность кода, соответствие принятым стандартам кодирования;

- корректность интеграции компонентов программных средств в систему;
- адекватность, полнота и непротиворечивость документации.

Аттестация (валидация) (validation process) – процесс проверки соответствия системы ожиданиям заказчика. Аттестация должна гарантировать соответствие программных средств спецификациям, требованиям и документации, возможность безопасного и надежного применения пользователем. Аттестацию рекомендуется выполнять путем тестирования.

Тестирование (testing process) – процесс обнаружения ошибок в программном приложении.

Качество программных средств (software quality) – совокупность характеристик программного обеспечения, удовлетворяющая установленным и предполагаемым потребностям.

Факторы, влияющие на качество программного обеспечения:

- функциональные – факторы, связанные с полнотой и удобством использования реализованных функций программного средства;
- административные – факторы, связанные с квалификацией персонала, организационной структурой и управлением персоналом;
- программно-архитектурные – факторы, связанные с процессом разработки программного обеспечения, выбранными методологиями, инструментальными средствами, архитектурой программного средства.

Надежность программных средств (software reliability) – способность безотказно выполнять функции при заданных условиях в течение заданного периода времени с достаточно большой вероятностью.

Отказ программного обеспечения (software failure) – проявление ошибки в программном средстве.

Ошибка (*software error*) – сбой, некорректное поведение программы.

Примитивами надежности программного обеспечения являются: завершенность, точность, автономность, устойчивость, защищенность.

Завершенность (*completeness*) – свойство, характеризующее степень обладания программного обеспечения необходимыми частями и чертами, требующимися для выполнения функций.

Точность (*accuracy*) – мера, характеризующая приемлемость величины погрешности в выдаваемых программами результатах с точки зрения предполагаемого использования.

Автономность (*self-containedness*) – свойство, характеризующее способность программного обеспечения выполнять предписанные функции без помощи или поддержки других компонент программного обеспечения.

Устойчивость (*robustness*) – свойство, характеризующее способность программного обеспечения продолжать корректное функционирование, несмотря на задание неправильных (ошибочных) входных данных.

Защищенность (*defensiveness*) – свойство, характеризующее способность программного обеспечения противостоять преднамеренным или нечаянным деструктивным (разрушающим) действиям пользователя.

Живучесть – свойство, характеризующее способность ПО продолжать корректное функционирование, несмотря на сбой частей программного обеспечения [1–5].

2 Место верификации в жизненном цикле разработки программного обеспечения

2.1 Процессы жизненного цикла программных средств

Жизненный цикл программного обеспечения (software life-cycle) – совокупность этапов, связанных с изменением программного обеспечения от исходных требований до окончания эксплуатации пользователем.

Модели жизненного цикла программных средств: каскадная, V-образная, спиральная.

Процессы жизненного цикла программных средств подразделяются на три группы:

- основные;
- вспомогательные;
- организационные.

Группа основных процессов включает процессы: приобретение, поставка, разработка, эксплуатация, сопровождение.

Группа вспомогательных процессов включает процессы: документирование, управление конфигурацией, обеспечение качества, верификация, аттестация, оценка, аудит, решение проблем.

Группа организационных процессов включает процессы: управление проектами, создание инфраструктуры проекта, определение, оценка и улучшение жизненного цикла, обучение [6].

2.2 Практическая работа № 1 «Жизненный цикл разработки и верификации программ»

Цель: закрепить теоретические знания и получить практические навыки в верификации программного приложения (проверка функциональных требований, предъявляемых к программному приложению, функций, архитектуры программы, программного кода).

Используемые приемы и технологии: технология визуального проектирования и событийного программирования, среда программирования Visual Studio 2019 Community, язык программирования Visual C++.

Ключевые термины: жизненный цикл, функциональные требования, спецификация программы, архитектура программы, программный код, интерфейс пользователя, входные и выходные данные, модуль программы.

Варианты заданий

Вариант 1

Сформулируйте цель программного приложения «Калькулятор» и функциональные требования к нему. Разработайте визуальное приложение на языке C++, формализующее работу калькулятора чисел. Калькулятор позволяет выполнять аддитивные и мультипликативные операции:

- сложение +;
- вычитание -;
- умножение *;
- деление /;
- остаток от деления %.

Выполните проверку функциональных требований к программе «Калькулятор», функций программы, архитектуры программы, программного кода.

Вариант 2

Сформулируйте цель программного приложения «Калькулятор сверхдлинных целых чисел» и функциональные требования к нему. Разработайте визуальное приложение на языке C++, формализующее работу калькулятора сверхдлинных целых чисел. Калькулятор позволяет выполнять:

1 аддитивные и мультипликативные операции:

- сложение +;
- вычитание -;
- умножение *;
- деление /,
- остаток от деления %.

2 операции сравнения сверхдлинных целых чисел:

- равно ==;
- меньше <;
- больше >;
- меньше или равно <=;
- больше или равно >=.

3 логические и побитовые операции:

- побитовое И(&);
- побитовое исключающее ИЛИ (^);
- побитовое ИЛИ (|);
- логическое ИЛИ (||);
- логическое И (&&).

4 операции определения наибольшего общего делителя (НОД) и наименьшего общего кратного (НОК).

Выполните проверку функциональных требований к программе «Калькулятор сверхдлинных целых чисел», функций программы, архитектуры программы, программного кода.

Методические указания

1 Сформулируйте цель программного приложения.
2 Определите функциональные требования к программному приложению.
3 Разработайте алгоритм программы.
4 Разработайте программное приложение.
5 Выберите модель жизненного цикла программного приложения.
6 Протестируйте программное приложение.
7 Оформите отчет по практической работе, включающий разделы:

- общее описание;
- требования к программному приложению;
- архитектура программы;
- тестирование системы: проверка программного кода, проверка архитектуры, проверка требований;
- код программы;
- скриншоты программы;
- выводы.

8 Ответьте на контрольные вопросы.

Контрольные вопросы

1 Что называется жизненным циклом программного обеспечения?
2 Какие модели жизненного цикла программных средств?
3 Какие основные процессы жизненного цикла программных средств?
4 Какие вспомогательные процессы жизненного цикла программных средств?
5 Какие организационные процессы жизненного цикла программных средств?

3 Оценка качества программных средств

3.1 Характеристики качества программного обеспечения

Модель качества программного обеспечения включает структурные составляющие:

- характеристики качества программного обеспечения;
- атрибуты;
- метрики;
- оценки значений атрибутов.

Характеристики качества программного обеспечения:

- функциональность – набор свойств, обуславливающий способность выполнять функции в соответствии с назначением;
- надёжность – набор свойств, обуславливающий способность сохранять работоспособность за определённый период времени;

- удобство – набор свойств, обуславливающий способность простого освоения программы, подготовки данных и внесение изменений в программу;
- эффективность – набор свойств, обуславливающий способность программного обеспечения обеспечивать требуемый уровень производительности в соответствии с выделенными ресурсами, временем и другими обозначенными условиями;
- сопровождаемость – набор свойств, обуславливающий способность адаптации к диагностике отказов и последствий внесения изменений;
- переносимость – набор свойств, обуславливающий способность адаптации к новой среде функционирования.

3.2 Практическая работа № 2
«Расчёт параметров лексического анализа текстов программ»

Цель: закрепить теоретические знания и получить практические навыки в определении оценок качества программного обеспечения на основе лексического анализа текстов программ.

Используемые приемы и технологии: технология визуального проектирования и событийного программирования, среда программирования Visual Studio 2019 Community, язык программирования Visual C++.

Ключевые термины: словарь операторов и операций, длина реализации программы, длина программы, объем программы, соотношение Холстеда, абсолютная сложность программы, относительная сложность программы, метрики Джилба, метрики Чепина.

Варианты заданий

Разработайте программу для расчёта значений функции. Оцените качество программы с использованием метрик Холстеда и Джилба.

Вариант 1

$$y = \begin{cases} a * x^2 - b * x + c, при\ x < 3, \text{b} \neq 0 \\ \dfrac{\text{x - a}}{\text{x - c}}, при\ x > 3\ и\ b = 0 \\ \dfrac{x}{c}, в\ остальных\ случаях \end{cases}$$

Вариант 2

$$y = \begin{cases} a * x^2 + b^2 * x, при\ a < 0, \text{x} \neq 0 \\ x - \dfrac{\text{a}}{\text{x - c}}, при\ a > 0\ и\ x = 0 \\ 1 + \dfrac{x}{c}, в\ остальных\ случаях \end{cases}$$

Вариант 3

$$y = \begin{cases} -a * x^2, при\ c < 0, \text{a} \neq 0 \\ \dfrac{\text{a - x}}{\text{c * x}}, при\ c > 0\ и\ a = 0 \\ \dfrac{x}{c}, в\ остальных\ случаях \end{cases}$$

Вариант 4

$$y = \begin{cases} -a * x^2 - b, при\ x < 5, \text{c} \neq 0 \\ \dfrac{\text{x - a}}{\text{x}}, при\ x > 5\ и\ c = 0 \\ \dfrac{-x}{c}, в\ остальных\ случаях \end{cases}$$

Вариант 5

$$y = \begin{cases} a * x^2 + b^2 * x, при\ c < 0, \text{b} \neq 0 \\ \dfrac{\text{x + a}}{\text{x + c}}, при\ c > 0\ и\ b = 0 \\ \dfrac{x}{c}, в\ остальных\ случаях \end{cases}$$

Вариант 6

$$y = \begin{cases} a - \dfrac{x}{10 + b}, при\ x < 0, \text{b} \neq 0 \\ \dfrac{\text{x - a}}{\text{x - c}}, при\ x > 0\ и\ b = 0 \\ 3 * x + \dfrac{2}{c}, в\ остальных\ случаях \end{cases}$$

Вариант 7

$$y = \begin{cases} -a * x - c, при\ c < 0, \text{x} \neq 0 \\ \dfrac{\text{x - a}}{\text{- c}}, при\ c > 0\ и\ x = 0 \\ \dfrac{b * x}{c - a}, в\ остальных\ случаях \end{cases}$$

Вариант 8

$$y=\begin{cases} a*x^2+b*x+c, при\ a<0, \mathrm{c}\neq 0 \\ \dfrac{-\mathrm{a}}{\mathrm{x}-\mathrm{c}}, при\ a>0\ и\ c=0 \\ a*(x+c), в\ остальных\ случаях \end{cases}$$

Вариант 9

$$y=\begin{cases} \dfrac{1}{a*x}-b, при\ x+5<0, \mathrm{c}=0 \\ \dfrac{\mathrm{x}-\mathrm{a}}{\mathrm{x}}, при\ x+5>0\ и\ c\neq 0 \\ \dfrac{10*x}{c-4}, в\ остальных\ случаях \end{cases}$$

Вариант 10

$$y=\begin{cases} a*x^2+b, при\ x+2<0, \mathrm{b}\neq 0 \\ \dfrac{\mathrm{x}-\mathrm{a}}{\mathrm{x}-\mathrm{c}}, при\ x+2>0\ и\ c\neq 0 \\ \dfrac{10*x}{c-4}, в\ остальных\ случаях \end{cases}$$

Разработайте программу на языке C++, формализующую расчет выходной переменной y функциональных зависимостей $y=f(x)$, заданных графическим способом (рисунки 3.1 – 3.10). Оцените качество программы с использованием метрик Холстеда и Джилба.

Вариант 1

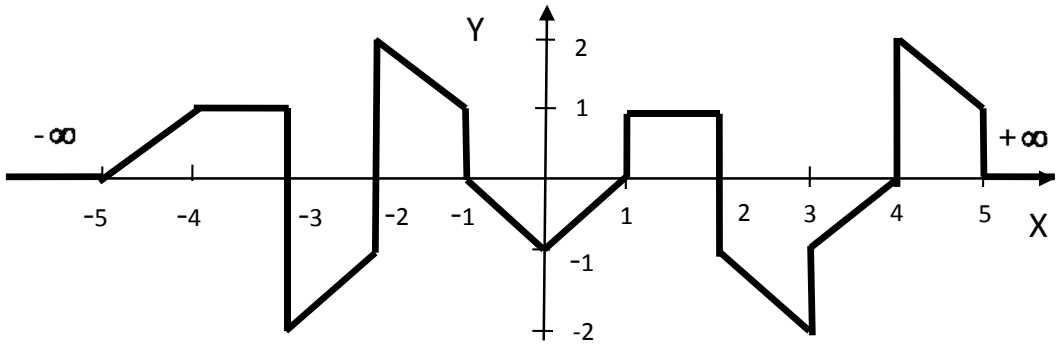

Рисунок 3.1 – Графический способ задания варианта 1

Вариант 2

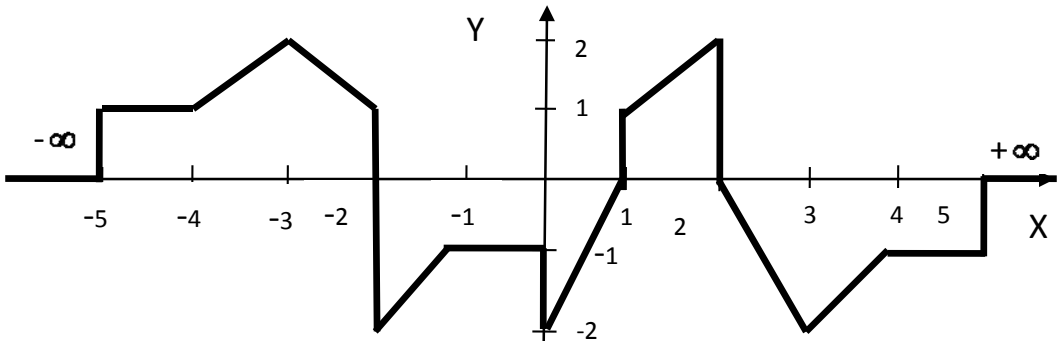

Рисунок 3.2 – Графический способ задания варианта 2

Вариант 3

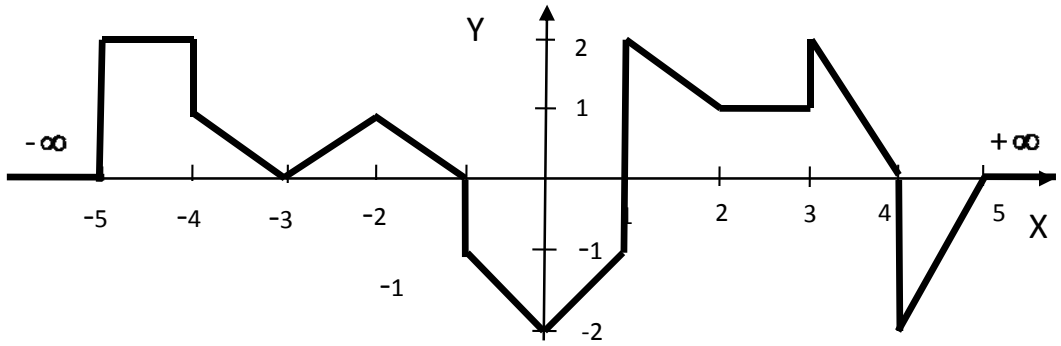

Рисунок 3.3 – Графический способ задания варианта 3

Вариант 4

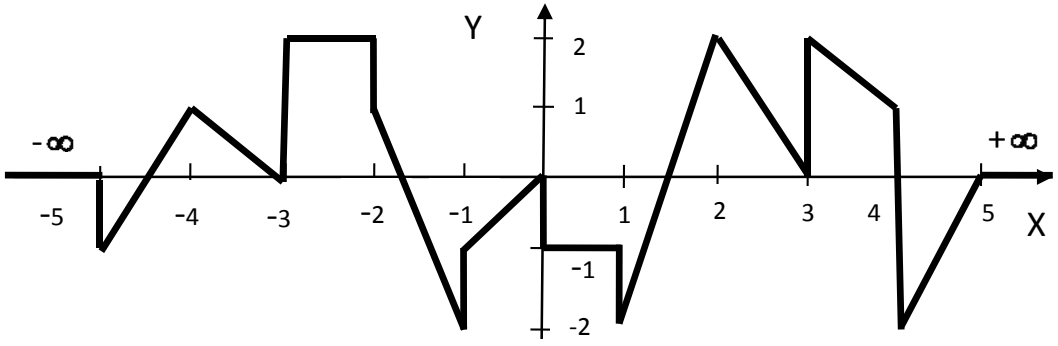

Рисунок 3.4 – Графический способ задания варианта 4

Вариант 5

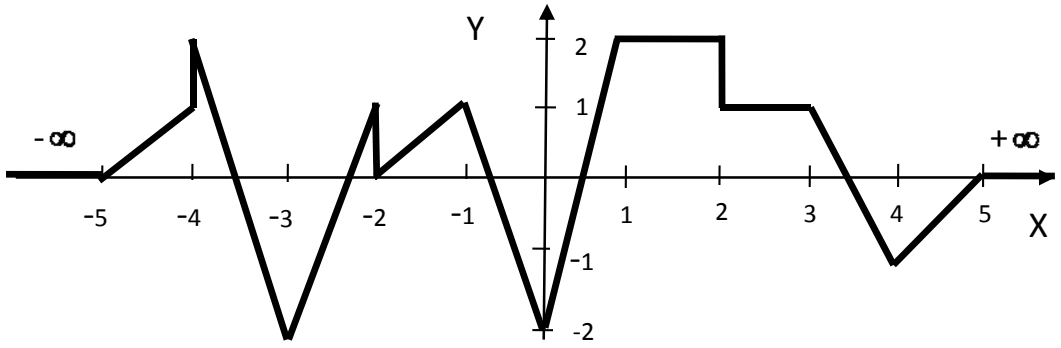

Рисунок 3.5 – Графический способ задания варианта 5

Вариант 6

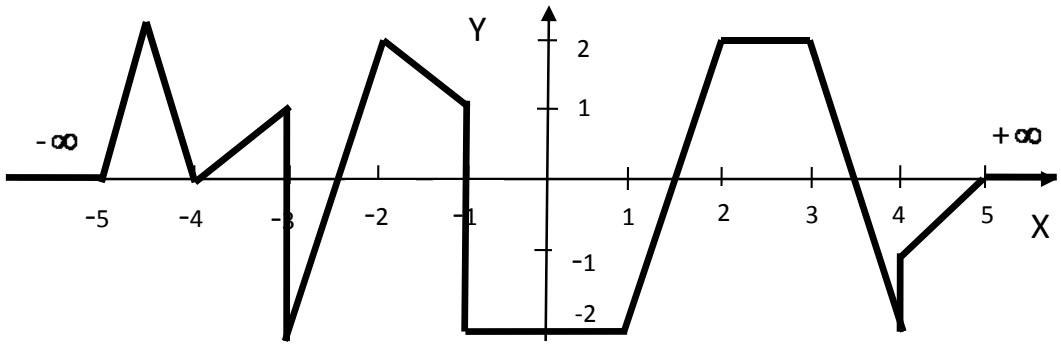

Рисунок 3.6 – Графический способ задания варианта 6

Вариант 7

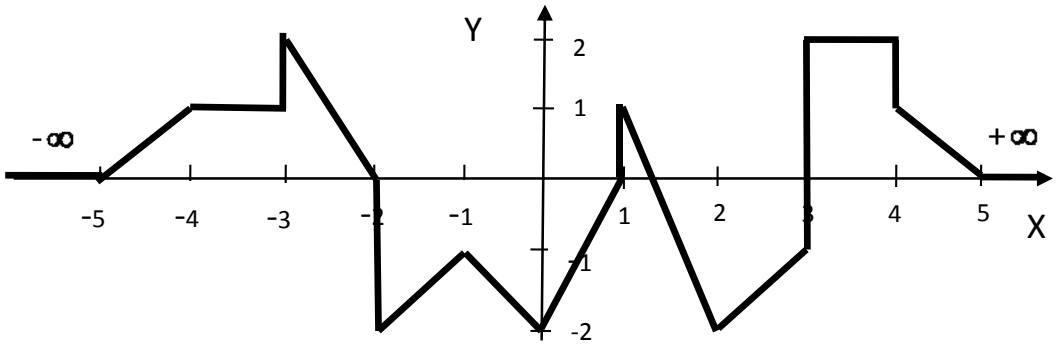

Рисунок 3.7 – Графический способ задания варианта 7

Вариант 8

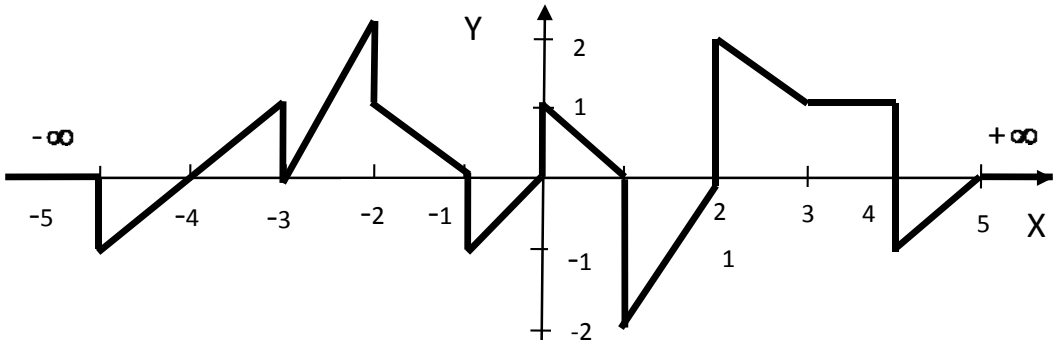

Рисунок 3.8 – Графический способ задания варианта 8

Вариант 9

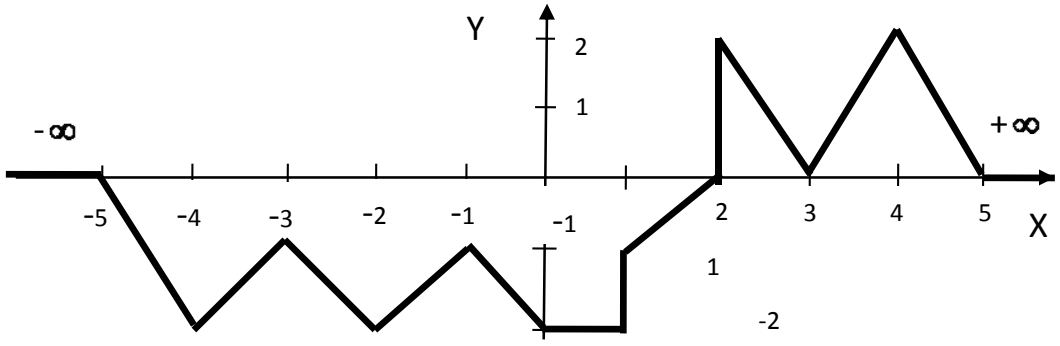

Рисунок 3.9 – Графический способ задания варианта 9

Вариант 10

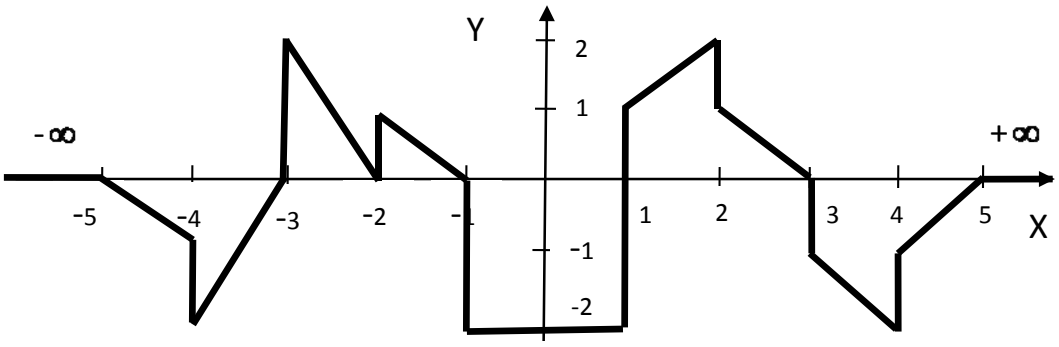

Рисунок 3.10 – Графический способ задания варианта 10

Разработайте программу обработки матрицы. На основе лексического анализа исходного текста программы оценить качество программы с использованием метрики Чепина.

Вариант 1. Дана целочисленная прямоугольная матрица $\|a_{n*m}\|$. Написать программу, определяющую величины:

1) суммы элементов в строках, которые содержат, хотя бы один отрицательный элемент;

2) минимальный элемент матрицы;

3) среднее арифметическое значение четных элементов строк и столбцов матрицы.

Средние арифметические значения четных элементов строк и столбцов матрицы упорядочить по убыванию методом вставки.

Вариант 2. Дана целочисленная квадратная матрица $\|a_{n*n}\|$. Написать программу, определяющую величины:

1) сумму элементов в строках, которые не содержат отрицательных элементов;

2) максимум среди элементов выше главной диагонали матрицы;

3) суммы четных элементов строк и столбцов матрицы.

Суммы четных элементов строк и столбцов матрицы упорядочить по убыванию методом обмена.

Вариант 3. Дана целочисленная квадратная матрица $\|a_{n*n}\|$. Написать программу, определяющую величины:

1) суммы элементов в столбцах, которые не содержат отрицательных элементов;

2) минимум среди модулей элементов ниже побочной диагонали матрицы;

3) суммы нечетных элементов строк и столбцов матрицы.

Суммы нечетных элементов строк и столбцов матрицы упорядочить по убыванию методом выбора.

Вариант 4. Дана целочисленная прямоугольная матрица $\|a_{n*m}\|$. Написать программу, определяющую величины:

1) количество ненулевых элементов матрицы;

2) максимальный и минимальный четные элементы столбцов матрицы;

3) суммы элементов матрицы, находящихся ниже побочной диагонали.

Элементы матрицы, находящиеся ниже побочной диагонали, упорядочить по убыванию методом обмена.

Вариант 5. Дана целочисленная прямоугольная матрица $\|a_{n*m}\|$. Написать программу, определяющую величины:

1) количество столбцов, содержащих, хотя бы один нулевой элемент;

2) номер строки, в которой находится самая длинная серия одинаковых элементов;

3) среднее геометрическое значение элементов строк и столбцов матрицы.

Средние геометрические значения элементов строк и столбцов матрицы упорядочить по возрастанию методом вставки.

Вариант 6. Дана целочисленная прямоугольная матрица $\|a_{n*m}\|$. Написать программу, определяющую величины:

1) количество нулевых элементов матрицы;

2) максимальный и минимальный нечетные элементы столбцов матрицы;

3) суммы элементов матрицы, находящихся выше побочной диагонали.

Элементы матрицы, находящиеся выше побочной диагонали, упорядочить по возрастанию методом вставки.

Вариант 7. Дана целочисленная прямоугольная матрица $\|a_{n*m}\|$. Написать программу, определяющую величины:

1) количество строк, не содержащих ни одного нулевого элемента;

2) максимальное значение из чисел, встречающихся в заданной матрице более одного раза;

3) суммы элементов строк и столбцов матрицы.

Суммы элементов строк и столбцов матрицы упорядочить по возрастанию методом обмена.

Вариант 8. Дана целочисленная прямоугольная матрица $\|a_{n*m}\|$. Написать программу, определяющую величины:

1) количество положительных элементов матрицы;

2) максимальный и минимальный нечетные элементы строк матрицы;

3) суммы элементов матрицы, находящихся ниже побочной диагонали.

Элементы матрицы, находящиеся ниже побочной диагонали, упорядочить по возрастанию методом выбора.

Вариант 9. Дана целочисленная прямоугольная матрица $\|a_{n*m}\|$. Написать программу, определяющую величины:

1) количество отрицательных элементов матрицы;

2) максимальный элемент матрицы;

3) суммы элементов матрицы, находящихся ниже главной диагонали.

Элементы матрицы, находящиеся выше главной диагонали, упорядочить по возрастанию методом обмена.

Вариант 10. Дана целочисленная прямоугольная матрица $\|a_{n*m}\|$. Написать программу, определяющую величины:

1) количество столбцов, не содержащих ни одного нулевого элемента;

2) максимальное и минимальное значение элементов матрицы;

3) среднее арифметическое значение элементов строк и столбцов матрицы.

Средние арифметические значения элементов строк и столбцов матрицы упорядочить по возрастанию методом выбора.

Методические указания

1 Разработайте алгоритм решения задачи.

2 Разработайте программное приложение.

3 Сформируйте словарь программы.

4 Рассчитайте характеристики программ на основе лексического анализа текста программного приложения:

- метрики Холстеда;
- метрики Джилба;
- метрики Чепина.

5 Оформите отчет по практической работе.

6 Ответьте на контрольные вопросы.

Контрольные вопросы

1 Что такое словарь программы?

2 Какие условия образования словаря программы?

3 Какие основные метрические характеристики программы?

4 Какой вид имеет соотношение Холстеда?

5 Как определяется потенциальный объем программы?

6 Как определяется уровень реализации программы?

7 Какие метрики программного обеспечения предложены Джилбом.

8 Какие виды переменных применяются при оценке качества программы с использованием метрик Чепина?

9 Как определяется метрика Чепина?

10 Для чего используются весовые коэффициенты в расчётном выражении значения метрики Чепина?

3.3 Практическая работа № 3
«Расчёт параметров структурной сложности программ»

Цель: закрепить теоретические знания и получить практические навыки в определении оценок структурной сложности программ.

Используемые приемы и технологии: технология визуального проектирования и событийного программирования, среда программирования Visual Studio 2019 Community, язык программирования Visual C++.

Ключевые термины: структурная сложность программы, вычислительный маршрут, маршрут принятия логических решений, основной маршрут тестирования Маккейба, граф потока управления, цикломатическая сложность, критерии выделения маршрутов, метрика Маккейба.

Варианты заданий

Вариант 1

В одномерном динамическом массиве, состоящем из *n* вещественных чисел, определите:

1) минимальный элемент массива;

2) сумму элементов массива, расположенных между первым и последним положительными элементами.

Преобразуйте массив таким образом, чтобы сначала располагались все элементы, равные нулю, а потом все остальные.

По разработанному алгоритму постройте граф потока управления и оцените алгоритмическую сложность программы.

Вариант 2

В одномерном динамическом массиве, состоящем из *n* целых чисел, определите:

1) максимальный элемент массива;

2) произведение элементов массива, расположенных между первым и вторым нулевыми элементами.

Преобразуйте массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в нечетных позициях, а во второй – элементы, стоявшие в четных позициях.

По разработанному алгоритму постройте граф потока управления и оцените алгоритмическую сложность программы.

Вариант 3

В одномерном динамическом массиве, состоящем из *n* целых чисел, упорядочите элементы массива по возрастанию простым методом вставки.

По разработанному алгоритму постройте граф потока управления и оцените алгоритмическую сложность программы.

Вариант 4

В одномерном динамическом массиве, состоящем из *n* вещественных чисел, упорядочите элементы массива по убыванию простым методом выбора.

По разработанному алгоритму постройте граф потока управления и оцените алгоритмическую сложность программы.

Вариант 5

В одномерном динамическом массиве, состоящем из *n* целых чисел, определите простые числа. Для простых чисел рассчитайте:

1) количество цифр;

2) среднее арифметическое значение.

По разработанному алгоритму постройте граф потока управления и оцените алгоритмическую сложность программы.

Вариант 6

В одномерном динамическом массиве, состоящем из *n* целых чисел, определите совершенные числа. Для совершенных чисел рассчитайте:

1) количество цифр;
2) среднее геометрическое значение.

По разработанному алгоритму постройте граф потока управления и оцените алгоритмическую сложность программы.

Вариант 7

Дана целочисленная квадратная матрица $\|a_{n*n}\|$. Разработайте программу, определяющую величины:

1) количество элементов матрицы, находящихся в диапазоне от A до B;
2) максимальный по модулю элемент матрицы.

По разработанному алгоритму постройте граф потока управления и оцените алгоритмическую сложность программы.

Вариант 8

Дана целочисленная квадратная матрица $\|a_{n*n}\|$. Разработайте программу, определяющую величины:

1) количество элементов матрицы, больших величины B;
2) суммы элементов, расположенных по периметру матрицы.

По разработанному алгоритму постройте граф потока управления и оцените алгоритмическую сложность программы.

Вариант 9

Разработайте программу на языке C++, формализующую определение попадания точки, заданной координатами x и y, в плоскую фигуру. По разработанному алгоритму постройте граф потока управления и оцените алгоритмическую сложность программы с использованием метрики Маккейба (рисунок 3.11).

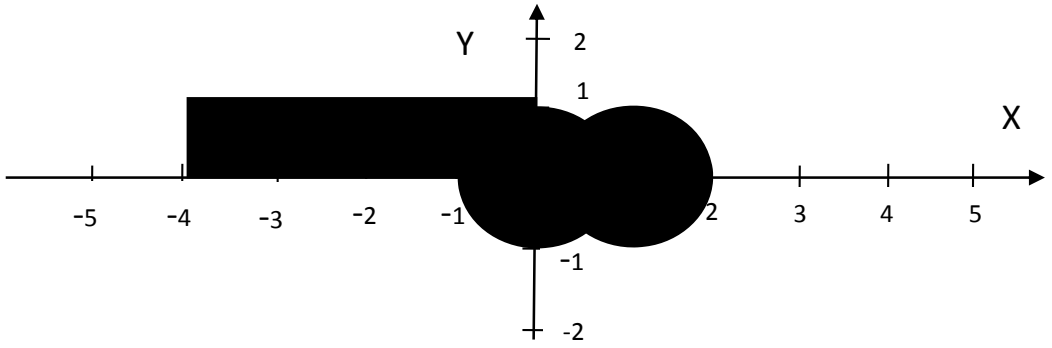

Рисунок 3.11 – Плоская фигура варианта 9

Вариант 10

Разработайте программу на языке C++, формализующую определение попадания точки, заданной координатами x и y, в плоскую фигуру. По разработанному алгоритму постройте граф потока управления и оцените алгоритмическую сложность программы с использованием метрики Маккейба (рисунок 3.12).

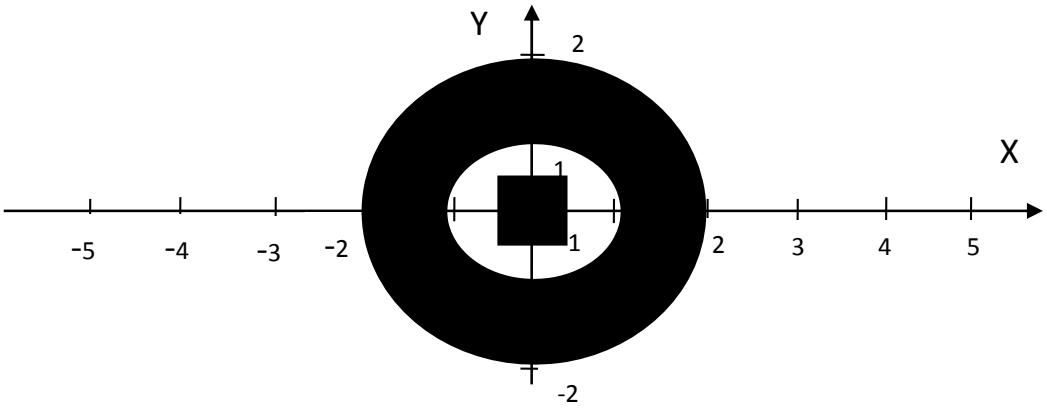

Рисунок 3.12 – Плоская фигура варианта 10

Методические указания

1 Разработайте алгоритм решения задачи.
2 Разработайте программное приложение.
3 Постройте граф потока управления.
4 Оцените алгоритмическую сложность:

- критерий 1;
- критерий 2;
- критерий 3;
- метрика Маккейба.

5 Оформите отчет по практической работе.
6 Ответьте на контрольные вопросы.

Контрольные вопросы

1 Что понимается под структурной сложностью программ?
2 Как рассчитывается сложность вычислительных маршрутов?
3 Как рассчитывается сложность маршрутов принятия логических решений?
4 Что определяют критерии 1, 2, 3?
5 Каким выражением определяется метрика Маккейба?

3.4 Практическая работа № 4 «Функционально-ориентированные метрики оценки качества программ»

Цель: закрепить теоретические знания и получить практические навыки в определении оценок программных средств на основе процедурно-ориентированных метрик (количество функциональных указателей, оценки связности и сцепления модулей в программе).

Используемые приемы и технологии: среда программирования Visual Studio 2019 Community, язык программирования Visual C++.

Ключевые термины: программный модуль, функциональные указатели, косвенные метрики, метрики свойств, указатели свойств, связность модулей.

Варианты заданий

Разработайте программу на языке C++ и оцените качество программы на основе применения функционально-ориентированных метрик: количество функциональных указателей, уровни связности и сцепления программных модулей.

Вариант 1. Дана целочисленная квадратная матрица $\|a_{n*n}\|$. Разработайте программу, определяющую величины:

1) количество элементов матрицы, содержащих простые числа;

2) минимум элементов, расположенных выше побочной диагонали;

3) суммы отрицательных элементов столбцов и строк матрицы.

Суммы отрицательных элементов столбцов и строк матрицы отсортируйте по убыванию методом вставки.

Вариант 2. Дана целочисленная квадратная матрица $\|a_{n*n}\|$. Разработайте программу, определяющую величины:

1) количество элементов матрицы, содержащих простые числа;

2) максимум среди модулей элементов, расположенных выше главной диагонали;

3) средние квадратичные значения четных столбцов и нечетных строк матрицы.

Средние квадратичные значения четных столбцов и нечетных строк матрицы отсортируйте по возрастанию методом обмена.

Вариант 3. Дана целочисленная квадратная матрица $\|a_{n*n}\|$. Разработайте программу, определяющую величины:

1) количество элементов матрицы, содержащих нулевые значения;

2) максимум элементов, расположенных ниже побочной диагонали;

3) суммы положительных элементов столбцов и строк матрицы.

Суммы положительных элементов столбцов и строк матрицы, отсортируйте по убыванию методом выбора.

Вариант 4. Дана целочисленная прямоугольная матрица $\|a_{n*m}\|$. Разработайте программу, определяющую величины:

1) количество элементов матрицы, содержащих простые числа;

2) минимум среди модулей элементов, расположенных ниже побочной диагонали;

3) средние квадратичные значения столбцов и строк матрицы.

Средние квадратичные значения столбцов и строк матрицы отсортируйте по убыванию методом вставки.

Вариант 5. Дана целочисленная квадратная матрица $\|a_{n*n}\|$. Разработайте программу, определяющую величины:

1) количество элементов матрицы, содержащих простые числа;

2) минимум среди элементов главной диагонали;

3) средние арифметические значения четных столбцов и нечетных строк матрицы.

Средние арифметические значения четных столбцов и нечетных строк матрицы отсортируйте по убыванию методом обмена.

Вариант 6. Дана целочисленная квадратная матрица $\|a_{n*n}\|$. Разработайте программу, определяющую величины:

1) количество элементов матрицы, находящихся в диапазоне от A до B;

2) максимальный по модулю элемент матрицы;

3) суммы элементов, расположенных на диагоналях матрицы.

Элементы, расположенные на диагоналях матрицы, отсортируйте по убыванию методом выбора.

Вариант 7. Дана целочисленная квадратная матрица $\|a_{n*n}\|$. Разработайте программу, определяющую величины:

1) количество элементов матрицы, содержащих простые числа;

2) максимум среди элементов побочной диагонали;

3) средние арифметические значения столбцов и строк матрицы.

Средние арифметические значения столбцов и строк матрицы отсортируйте по возрастанию методом вставки.

Вариант 8. Дана целочисленная прямоугольная матрица $\|a_{n*m}\|$. Разработайте программу, определяющую величины:

1) количество элементов матрицы, больших величины B;

2) минимальный по модулю элемент матрицы;

3) суммы элементов, расположенных по периметру матрицы.

Элементы, расположенные по периметру матрицы, отсортируйте по убыванию методом обмена.

Вариант 9. Дана целочисленная квадратная матрица $\|a_{n*n}\|$. Разработайте программу, определяющую величины:

1) количество элементов матрицы, содержащих простые числа;

2) минимум среди элементов, расположенных выше побочной диагонали;

3) средние квадратичные значения нечетных столбцов и четных строк матрицы.

Средние квадратичные значения нечетных столбцов и четных строк матрицы отсортируйте по возрастанию методом выбора.

Вариант 10. Дана целочисленная прямоугольная матрица $\|a_{n*m}\|$. Разработайте программу, определяющую величины:

1) количество элементов матрицы, меньших величины B;

2) максимальный и минимальный нечетные элементы столбцов матрицы;

3) суммы элементов матрицы, находящихся выше побочной диагонали.

Элементы матрицы, находящиеся выше побочной диагонали, отсортируйте по возрастанию методом вставки.

Методические указания

1 Разработайте алгоритм решения задачи.
2 Разработайте программное приложение.
3 Рассчитайте количество функциональных указателей:

- определите общее количество функциональных указателей;
- определите значения коэффициентов регулировки сложности;
- рассчитайте сумму коэффициентов;

4 Рассчитайте показатели качества, производительности, удельной стоимости и документированности.
5 Определите уровень связности.
6 Определите уровень сцепления.
7 Оформите отчет по практической работе.
8 Ответьте на контрольные вопросы.

Контрольные вопросы

1 Что понимается под функциональным указателем?
2 По какому выражению рассчитывается количество функциональных указателей?
3 Каким образом рассчитываются косвенные метрики: производительность, качество, удельная стоимость, документированность?
4 Что такое метрики свойств?
5 Как рассчитываются указатели свойств?
6 Что представляет собой связность модулей?
7 Какие типы связности модулей существуют?
8 Какие факторы характеризуют взаимодействие программного модуля с остальной частью программы?
9 Какая процедура определения типа связности?
10 Что понимается под сцеплением модулей программы?
11 Какие типы сцепления модулей существуют?
12 Какие методы и способы снижения степени сцепления программных модулей существуют?
13 Какие преимущества достигаются при слабом сцеплении программных модулей?

3.5 Практическая работа № 5 «Расчёт параметров качества программ при объектно-ориентированном программировании»

Цель: закрепить теоретические знания и получить практические навыки в определении оценок программных средств на основе объектно-ориентированных метрик: Мартина, Чидамбера и Кемерера, Лоренца и Кидда, Абреу.

Используемые приемы и технологии: среда программирования Visual Studio 2019 Community, язык программирования Visual C++.

Ключевые термины: категория классов, метрики Мартина, метрики Чидамбера и Кемерера: взвешенные методы на класс, количество методов на класс, глубина дерева наследования, связность между классами объектов, количество откликов на класс, отсутствие сцепления в методах, метрики Лоренца и Кидда.

Варианты заданий

Разработайте программу на языке C++, формирующую список объектов, и определите оценки характеристик программы на основе метрик Мартина, Чидамбера и Кемерера, Лоренца и Кидда, Абреу.

Вариант 1. Создайте базовый класс **Point** (Точка), характеризующийся двумя координатами x и y. Создайте производные классы **Circle** (Окружность) и **Ellipse** (Эллипс) от базового класса **Point**. Для класса **Circle** определите методы расчёта площади и длины окружности и вывода информации на экран. Для класса **Ellipse** определите методы расчёта площади и периметра эллипса и вывода информации на экран.

Вариант 2. Создайте базовый класс **Triangle** (Треугольник) с полями-сторонами. Определите методы изменения сторон и вычисления углов, периметра и площади треугольника. Создайте производные классы **RightTriangle** (Прямоугольный треугольник), **IsoscelesTriangle** (Равнобедренный треугольник), **EquilateralTriangle** (Равносторонний треугольник) от базового класса **Triangle**. Определите методы расчета периметра, площади треугольников и вывода информации на экран.

Вариант 3. Создайте базовый класс **Parallelogram** (Параллелограмм), который характеризуется длинами двух сторон и углом между ними. Для класса определите методы, вычисляющие периметр и площадь параллелограмма и вывода информации на экран. Создайте производные классы **Rectangle** (Прямоугольник) и **IsoscelesTrapezium** (Равнобедренная трапеция) от базового класса **Parallelogram**. Для класса **Rectangle** определите функции-члены, вычисляющие периметр, площадь, радиус описанной около прямоугольника окружности и вывода информации на экран. Для класса **IsoscelesTrapezium** определите функции-члены, вычисляющие высоту, диагонали, периметр, площадь (формула Герона) трапеции и вывода информации на экран.

Вариант 4. Создайте базовый класс **Person** (Человек), имеющий поля: фамилия, имя, отчество, пол, год рождения, вес, рост. Определите функции-члены, рассчитывающие идеальный вес человека по формуле Лоренца ($m_{идеал\_ж} = h_{ж} - 100 - \frac{h_{ж} - 150}{2}$, где $h_{ж}$ – рост женщины, см; $m_{идеал\_м} = h_{м} - 100 - \frac{h_{м} - 150}{4}$, где $h_{м}$ – рост мужчины, см), показатель нормальной массы тела Брейтмана ($m_{норм\_масса} = h * 0.7 - 50$, где h – рост

человека, см), индекс Кетле ($I_{массы\_тела} = \frac{m}{h*h}$. где m – вес человека, кг; h – рост человека, м) и выведите информацию на экран. Создайте производные классы **Student** (Студент) и **Undergraduate** (Магистрант) от базового класса **Person.** Класс **Student** имеет дополнительные поля: номер зачётной книжки, номер семестра, массив средних оценок за семестры, номер группы. Для производного класса **Student** определите методы расчета итоговой средней оценки за период обучения и вывода информации. Класс **Undergraduate** имеет дополнительные поля: наименование направления обучения, тема выпускной квалификационной работы. Для класса **Undergraduate** определите методы вывода информации на экран.

Вариант 5. Создайте базовый класс **Car** (Легковой автомобиль), характеризуемый торговой маркой, числом цилиндров, мощностью. Определите методы расчёта полной массы легкового автомобиля ($m_{лег\_ав} = m_0 + 75 * n_{пас} + m_б$, где m_0 – масса автомобиля, кг; $n_{пас}$ – число пассажиров; $m_б$ – масса багажа, кг), и выведите информацию на экран. Создайте производные классы **Lorry** (Грузовой автомобиль) и **Bus** (Городской автобус) от базового класса **Car**. Класс **Lorry** имеет дополнительные поля: грузоподъёмность, запас хода. Определите методы расчёта полной массы грузового автомобиля ($m_{гр\_ав} = m_0 + 75 * n_{пас} + m_{гр}$, где $m_{гр}$ – грузоподъёмность автомобиля, кг) и выведите информации на экран. Класс **Bus** имеет дополнительные поля: число мест для проезда сидя, число мест для проезда стоя, максимальная скорость, минимальный радиус поворота. Определите методы расчёта полной массы городского автобуса ($m_{гор\_ав} = m_0 + 75 * (n_{сид} + n_{ст} + 2)$, где $n_{сид}$ – число мест для проезда сидя; $n_{ст}$ – число мест для проезда стоя) и выведите информацию на экран.

Вариант 6. Создайте базовый класс **ComplexNumber** (Комплексное число) для работы с комплексными числами. Комплексное число представляется парой чисел (a,b), где a – действительная часть, b – мнимая часть. Определите методы сложения **add**: $(a,b)+(c,d)=(a+c,b+d)$, вычитания **sub**: $(a,b)-(c,d)=(a-c,b-d)$, умножения **mul**: $(a,b)*(c,d)=((a*c-b*d),(a*d+b*c))$, деления **div:** $(a,b)/(c,d)=((a*c+b*d)/(c*c+d*d),(b*c-a*d)/(c*c+d*d))$, сравнение **equ**: $(a,b)=(c,d)$, если $(a=c)$ и $(b=d)$. Создайте производный класс **ComplexDerived** от базового класса **ComplexNumber**. Определите методы расчёта модуля комплексного числа **modul:** $\sqrt{a^2+b^2}$, комплексно сопряженного числа **conj**: $conj(a,b)=(a,-b)$ и вывода данных.

Вариант 7. Создайте класс **GasPayment** (Платёж за газ), включающий поля: фамилия, имя, отчество, количество потреблённого газа (куб. м), тариф. Определите методы расчёта суммы оплаты, формирование ведомости оплаты

(месяц, фамилия, имя, отчество, количество газа, тариф, сумма), формирование файла данных ведомостей оплаты, вывод данных.

Вариант 8. Создайте класс **PhonePayment** (Платёж за телефон), включающий поля: фамилия, имя, отчество, время разговора (мин), тариф. Определите методы расчёта суммы оплаты, формирование ведомости оплаты (месяц, фамилия, имя, отчество, время разговора, тариф, сумма), формирование файла данных ведомостей оплаты, вывод данных.

Вариант 9. Создайте класс **InternetPayment** (Платёж за Интернет), включающий поля: фамилия, имя, отчество, трафик (Мбайт), тариф. Определите методы расчёта суммы оплаты, формирование ведомости оплаты (месяц, фамилия, имя, отчество, трафик, тариф, сумма), формирование файла данных ведомостей оплаты, вывод данных.

Вариант 10. Создайте класс **WaterPayment** (Платёж за воду), включающий поля: фамилия, имя, отчество, количество потреблённой воды (куб. м), тариф. Определите методы расчёта суммы оплаты, формирование ведомости оплаты (месяц, фамилия, имя, отчество, количество воды, сумма), формирование файла данных ведомостей оплаты, вывод данных.

Методические указания

1 Разработайте алгоритм решения задачи.

2 Разработайте программное приложение.

3 Определите оценки характеристик программы на основе объектно-ориентированных метрик:

- Мартина;
- Чидамбера и Кемерера;
- Лоренца и Кидда;
- Абреу.

4 Оформите отчет по практической работе.

5 Ответьте на контрольные вопросы.

Контрольные вопросы

1 Какие условия существования связности группы классов?

2 Какие метрики Мартина?

3 Что позволяет оценить абстрактность категории?

4 Что называется главной последовательностью?

5 Какие группы метрик Лоренца и Кидда?

6 Какие метрики Лоренца и Кидда?

7 Какие цели достигаются применением метрик Абреу?

8 Какие метрики Абреу?

4 Оценка надежности программных средств

4.1 Показатели надежности программных средств

Факторы, влияющие на оценку надёжности программных средств:

1 программные методы и средства технологии программирования, применяемые в процессах разработки способствующие достижению требуемой надежности.

2 тестирование и проверка функционирования созданного программного средства со сбором данных о результатах обнаружения ошибок и интенсивности отказов в интервалах времени функционирования.

Показатели надежности:

- $P(t)$ – вероятность того, что ни одна ошибка не появится на интервале от 0 до t (функция надежности):

$$P(t) = P(T \geq t) \tag{4.1}$$

- $Q(t)$ – вероятность того, что ошибка появится на интервале от 0 до t (функция отказов):

$$Q(t) = 1 - P(t) \tag{4.2}$$

- $f(t)$ – плотность распределения времени безотказной работы программного обеспечения:

$$f(t) = Q'(t) = -P'(t) \tag{4.3}$$

- $\lambda(t)$ – интенсивность отказов или условная вероятность того, что ошибка появится на интервале от 0 до $t + \Delta t$ при условии, что на интервале от 0 до t ошибок не было (функция риска):

$$\lambda(t) = -\frac{P'(t)}{P(t)} = \frac{f(t)}{P(t)}\ , \qquad P(t) = \exp(-\int_0^t \lambda(u)du) \tag{4.4}$$

- T_{cp} – среднее время между отказами:

$$T_{cp} = \int_0^\infty P(u)du = \int_0^\infty u * f(u)du. \tag{4.5}$$

4.2 Модели надежности программных средств

Модель надежности программного обеспечения относится к математической модели, построенной для оценки зависимости надежности программного обеспечения от параметров. Классификационная схема моделей надежности приведена на рисунке 4.1 [5].

4.2.1 Динамическая модель надежности

Модель Шумана позволяет определить вероятность безотказной работы на интервале времени от 0 до t. Модель Шумана основана на следующих допущениях:

- общее число команд в программе на машинном языке постоянно;
- в начале компоновочных испытаний число ошибок равно некоторой постоянной величине, и по мере исправления ошибок их становится меньше. В ходе испытаний программы новые ошибки не вносятся;
- ошибки изначально различимы, по суммарному числу исправленных ошибок можно судить об оставшихся;
- интенсивность отказов программы пропорциональна числу остаточных ошибок.

Предполагается, что до начала тестирования (т.е. в момент $t = 0$) имеется M ошибок. В течение времени тестирования τ обнаруживается $\varepsilon_1(t)$ ошибок в расчете на одну команду в машинном языке.

Тогда удельное число ошибок на одну машинную команду, оставшихся в системе после времени тестирования τ, равно

$$\varepsilon_2(\tau) = \frac{M}{I} - \varepsilon_1(\tau), \qquad (4.6)$$

где I – общее число машинных команд, которое предполагается постоянным в рамках этапа тестирования;

M – количество ошибок до начала тестирования программы;

$\varepsilon_1(\tau)$ – количество ошибок в расчёте на команду в машинном языке, обнаруженных при тестировании за время τ.

Предполагается, что значение функции количества ошибок $Z(t)$ пропорционально числу ошибок, оставшихся в программе после израсходованного на тестирование времени τ.

$$Z(t) = C * \varepsilon_2(\tau), \qquad (4.7)$$

где C – коэффициент пропорциональности;

t – время работы программы без отказов;

$\varepsilon_2(\tau)$ – удельное число ошибок на одну машинную команду, оставшихся в программе после времени тестирования τ.

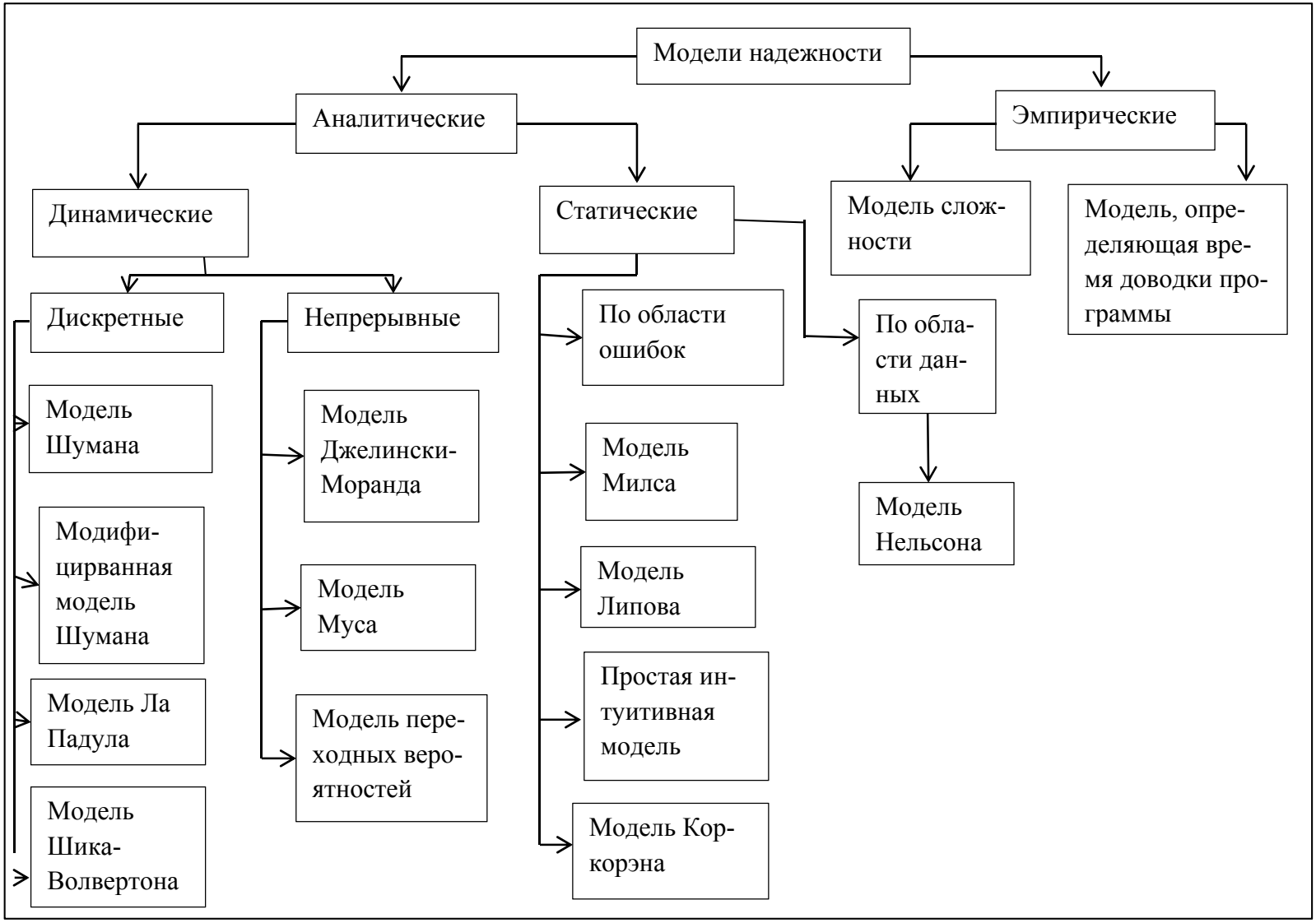

Рисунок 4.1 – Классификационная схема моделей надежности программного обеспечения

Тогда, если время работы программы без отказа t отсчитывается от точки $t = 0$, а τ остается фиксированным, функция надежности или вероятность безотказной работы на интервале от 0 до t, определяется по формуле

$$P(t,\tau) = \exp(-C * (\frac{M}{I} - \varepsilon_1(\tau)) * t) \quad (4.8)$$

Среднее время определяется по формуле

$$t_{cp} = \frac{1}{C * (\frac{M}{I} - \varepsilon_1(\tau))} \quad (4.9)$$

Необходимо найти начальное значение ошибок M и коэффициент пропорциональности C. Эти неизвестные оцениваются путем пропуска функционального теста в двух точках переменной оси отладки t_a и t_b, выбранных так, что $\varepsilon_1(t_a) < \varepsilon_1(t_b)$.

В процессе тестирования собирается информация о времени и количестве ошибок на каждом прогоне, т. е. общее время тестирования τ складывается из времени каждого прогона

$$\tau = \tau_1 + \tau_2 + \tau_3 + ... + \tau_n \quad (4.10)$$

Предполагая, что интенсивность появления ошибок постоянна и равна λ, можно вычислить ее как число ошибок в единицу времени:

$$\lambda = \frac{\sum_{i=1}^{n} A_i}{\tau}, \quad (4.11)$$

где A_i – количество ошибок на i-ом прогоне;

τ – общее время тестирования программы;

i – номер прогона, $i = \overline{1,n}$;

n – количество прогонов.

Среднее время определяется по формуле

$$t_{cp} = \frac{\tau}{\sum_{i=1}^{n} A_i} \quad (4.12)$$

Имея данные для двух различных моментов тестирования t_a и t_b, можно сопоставить уравнения 4.9 и 4.11 при τ_a и τ_b:

$$\frac{1}{\lambda_a} = \frac{1}{C * (\frac{M}{I} - \varepsilon_1(\tau_a))} \qquad (4.13)$$

$$\frac{1}{\lambda_b} = \frac{1}{C * (\frac{M}{I} - \varepsilon_1(\tau_b))} \qquad (4.14)$$

Из соотношений (4.13) и (4.18) найдем неизвестные параметры C и M :

$$M^* = I * \frac{\frac{\lambda_b}{\lambda_a} * \varepsilon_1(\tau_a) - \varepsilon_1(\tau_b)}{\frac{\lambda_b}{\lambda_a} - 1} \qquad (4.15)$$

$$C^* = \frac{\lambda_a}{\frac{M^*}{I} - \varepsilon_1(\tau_a)} \qquad (4.16)$$

Получив неизвестные M^* и C^*, можно рассчитать надежность программы по формуле 4.8.

Например, программа содержит 2 000 командных строк, из них, до начала эксплуатации (после периода отладки) 15 командных строк содержат ошибки. После 20 дней работы обнаружена 1 ошибка. Необходимо найти среднее время безошибочной работы программы и интенсивность отказов программы при коэффициенте пропорциональности, равном 0,7.

$I = 2000$ – количество командных строк;

$M = 15$ – количество командных строк, содержащих ошибки;

$t = 20$ – время работы, дни;

$\tau = 1$ – количество обнаруженных ошибок.

$$\varepsilon_2(\tau) = \frac{M}{I} - \varepsilon_1(\tau)$$

$$P(t,\tau) = \exp(-C * (\frac{M}{I} - \varepsilon_1(\tau)) * t)$$

$$C = 0.7$$

$$\varepsilon_1 = \frac{1}{2000} = 0.0005$$

$$\varepsilon_2 = \frac{151}{2000} - 0.0005 = 0.007$$

$P(t) = \exp(-0.7 * (\frac{15}{2000} - 0.0005) * 20) = 0.90661$ – функция надежности;

$$t_{cp} = \frac{1}{0.7 * (\frac{15}{2000} - 0.0005)};$$

$\lambda = 0.0049$ – интенсивность появления ошибок.

Вероятность безошибочной работы программы в течение 90 суток рассчитывается по формуле

$$P(t) = \exp(-0.7 * (\frac{15}{2000} - 0.0005) * 90) = 0.643393 \text{ [5-9]}$$

4.2.2 Статическая модель надёжности по области ошибок

Модель Миллса позволяет определить количество ошибок до начала тестирования и степень отлаженности программы.

Использование этой модели предполагает необходимость перед началом тестирования искусственно вносить в программу ошибки. Ошибки вносятся случайным образом и фиксируются в протоколе искусственных ошибок. Специалист, проводящий тестирование, не знает ни количества, ни характера внесенных ошибок. Предполагается, что все ошибки (естественные и искусственные) имеют равную вероятность быть найденными в процессе тестирования.

Программа тестируется в течение некоторого времени, и собирается статистика об обнаруженных ошибках.

Пусть n – количество собственных ошибок программы, обнаруженных после тестирования, и ν – количество искусственно внесенных ошибок. Тогда первоначальное число ошибок в программе N оценивается по формуле Миллса

$$N = n * \frac{S}{\nu}, \qquad (4.17)$$

где S – количество искусственно внесенных ошибок.

Вторая часть модели связана с проверкой гипотезы об N_0. Пусть в программе первоначально N_0 ошибок. Вносим искусственно в программу S ошибок и тестируем её до тех пор, пока все искусственно внесенные ошибки не будут обнаружены. Пусть при этом обнаружено n собственных ошибок программы. Вероятность, что в программе первоначально было N_0 ошибок, можно рассчитать по выражению

$$P = \begin{cases} 0, & n > N_0; \\ \dfrac{S}{S + N_0 + 1}, & n \le N_0 \end{cases} \qquad (4.18)$$

Если обнаружено только ν искусственно внесенных ошибок, то применяют выражение

$$P = \begin{cases} 0, & n > N_0; \\ \dfrac{C_S^{\nu-1}}{C_{S+K+1}^{N_0+\nu}}, & n \le N_0, \end{cases} \qquad (4.19)$$

где $C_n^m = \dfrac{n!}{m!*(n-m)!}$ – число сочетаний из n элементов по m.

Достоинства модели Миллса:

- простота применяемого математического аппарата;
- наглядность.

Недостатки модели Миллса:

- необходимость внесения искусственных ошибок (этот процесс плохо формализуем);
- величина *K* основывается на интуиции и опыте человека, производящего оценку [5, 6].

4.2.3 Статическая модель надёжности по области данных

Модель Нельсона позволяет определить вероятность события, при котором прогон программы на заданном наборе исходных данных не приведёт к отказу работы программы.

Надёжность программного обеспечения рассчитывается по формуле

$$P = 1 - \sum_{i=1}^{k} \frac{m_i}{M_i} * p_i, \qquad (4.20)$$

где M_i – число прогонов программы, $i = 1,k$;

m_i – число прогонов программы с отказом, $i = \overline{1,k}$;

p_i – вероятность выполнения программы на i наборе входных данных;

i – номер программы, $i = \overline{1,k}$;

k – число непересекающихся областей входных данных.

Модель Нельсона применяется для расчёта надёжности программы на всех стадиях жизненного цикла.

4.3 Практическая работа № 6 «Аналитическая динамическая модель надежности. Модель Шумана»

Цель: закрепить теоретические знания и получить практические навыки в разработке программы оценки показателей качества с применением модели Шумана.

Используемые приемы и технологии: технология визуального проектирования и событийного программирования, среда программирования Visual Studio 2019 Community, язык программирования Visual C++.

Ключевые термины: надежность программного обеспечения, показатели надежности, модель Шумана.

Варианты заданий

Вариант 1. В одномерном массиве, состоящем из *n* вещественных элементов, определите:

1) количество элементов массива больших C;

2) произведение элементов массива, расположенных после максимального по модулю элемента.

Преобразуйте массив таким образом, чтобы сначала располагались все отрицательные элементы, а потом — все положительные (элементы, равные нулю, считать положительными).

Вариант 2. В одномерном массиве, состоящем из *n* вещественных элементов, определите:

1) количество отрицательных элементов массива;

2) сумму модулей элементов массива, расположенных после минимального по модулю элемента.

Замените все отрицательные элементы массива их квадратами и отсортируйте элементы массива по возрастанию.

Вариант 3. В одномерном массиве, состоящем из *n* целых элементов, определите:

1) количество положительных элементов массива;

2) сумму элементов массива, расположенных после последнего элемента, равного нулю.

Преобразуйте массив таким образом, чтобы сначала располагались все элементы, целая часть которых не превышает 1, а потом — все остальные.

Вариант 4. В одномерном массиве, состоящем из *n* вещественных элементов, определите:

1) количество элементов массива меньших C;

2) сумму целых частей элементов массива, расположенных после последнего отрицательного элемента.

Преобразуйте массив таким образом, чтобы сначала располагались все элементы, отличающиеся от максимального не более, чем на 20 %, а потом – все остальные.

Вариант 5. В одномерном массиве, состоящем из *n* вещественных элементов, определите:

1) произведение отрицательных элементов массива;

2) сумму положительных элементов массива, расположенных до максимального элемента.

Измените порядок следования элементов в массиве на обратный.

Вариант 6. В одномерном массиве, состоящем из *n* вещественных элементов, определите:

1) произведение положительных элементов массива;

2) сумму элементов массива, расположенных до минимального элемента.

Упорядочите по возрастанию отдельно элементы, стоящие на четных местах, и элементы, стоящие на нечетных местах.

Вариант 7. В одномерном массиве, состоящем из *n* вещественных элементов, определите:

1) среднее арифметическое элементов, расположенных до первого и после последнего нулевых значений;

2) поменяйте местами первый и максимальный элементы, последний и минимальный элементы.

Упорядочите по убыванию отдельно элементы, стоящие на четных местах, и элементы, стоящие на нечетных местах.

Вариант 8. В одномерном массиве, состоящем из *n* целых элементов, определите:

1) произведение элементов, находящихся между минимальным и максимальным элементами массива;

2) удалите из массива простое число.

Упорядочите по возрастанию элементы массива.

Вариант 9. В одномерном массиве, состоящем из *n* вещественных элементов, определите:

1) произведение отрицательных элементов массива;

2) сумму элементов массива, расположенных до максимального элемента.

Упорядочите по возрастанию отдельно элементы, стоящие на четных местах, и элементы, стоящие на нечетных местах.

Вариант 10. В одномерном массиве, состоящем из *n* вещественных элементов, определите:

1) среднее значение элементов, расположенных в массиве между первым последним нулевыми элементами;

2) поменяйте местами максимальный и минимальный элементы.

Упорядочите по возрастанию отдельно элементы, стоящие на четных местах, и элементы, стоящие на нечетных местах.

Методические указания

1 Разработайте алгоритм программы.

2 Разработайте программное приложение.

3 Рассчитайте оценки надёжности программы.

4 Оформите отчет по практической работе, включающий разделы:

- постановка задачи;
- блок-схема программы;
- расчётные данные;
- скриншоты программы;

- код программы;
- выводы.

5 Ответьте на контрольные вопросы.

Контрольные вопросы

1 Что позволяет определить модель Шумана?
2 Какие условия используются в модели Шумана?
3 По какой формуле рассчитывается удельное число ошибок?
4 Какой вид имеет функция надёжности модели Шумана?
5 Каким образом определяется начальное значение ошибок?
6 Каким образом рассчитывается коэффициент пропорциональности?
7 Какие преимущества модели Шумана?
8 Какие недостатки модели Шумана?

4.4 Практическая работа № 7 «Аналитическая статическая модель надежности. Модель Миллса»

Цель: закрепить теоретические знания и получить практические навыки в разработке программы оценки показателей качества с применением модели Миллса.

Используемые приемы и технологии: технология визуального проектирования и событийного программирования, среда программирования Visual Studio 2019 Community, язык программирования Visual C++.

Ключевые термины: надежность программного обеспечения, показатели надежности, модель Миллса.

Варианты заданий

Вариант 1. Дана целочисленная прямоугольная матрица $\|a_{n*m}\|$. Разработайте программу, определяющую величины:

1) количество строк, не содержащих ни одного нулевого элемента;

2) максимальное значение из чисел, встречающихся в заданной матрице более одного раза;

3) суммы элементов строк и столбцов матрицы.

Суммы элементов строк и столбцов матрицы отсортируйте по возрастанию методом обмена.

Вариант 2. Дана целочисленная прямоугольная матрица $\|a_{n*m}\|$. Разработайте программу, определяющую величины:

1) количество столбцов, не содержащих ни одного нулевого элемента;

2) максимальное и минимальное значение элементов матрицы;

3) среднее арифметическое значение элементов строк и столбцов матрицы.

Средние арифметические значения элементов строк и столбцов матрицы отсортируйте по возрастанию методом выбора.

Вариант 3. Дана целочисленная прямоугольная матрица $\|a_{n*m}\|$. Разработайте программу, определяющую величины:

1) количество столбцов, содержащих, хотя бы один нулевой элемент;

2) номер строки, в которой находится самая длинная серия одинаковых элементов;

3) среднее геометрическое значение элементов строк и столбцов матрицы.

Средние геометрические значения элементов строк и столбцов матрицы отсортируйте по возрастанию методом вставки.

Вариант 4. Дана целочисленная квадратная матрица $\|a_{n*n}\|$. Разработайте программу, определяющую величины:

1) сумму элементов в строках, которые не содержат отрицательных элементов;

2) максимум среди элементов выше главной диагонали матрицы;

3) суммы четных элементов строк и столбцов матрицы.

Суммы четных элементов строк и столбцов матрицы отсортируйте по убыванию методом обмена.

Вариант 5. Дана целочисленная квадратная матрица $\|a_{n*n}\|$. Разработайте программу, определяющую величины:

1) суммы элементов в столбцах, которые не содержат отрицательных элементов;

2) минимум среди модулей элементов ниже побочной диагонали матрицы;

3) суммы нечетных элементов строк и столбцов матрицы.

Суммы нечетных элементов строк и столбцов матрицы отсортируйте по убыванию методом выбора.

Вариант 6. Дана целочисленная прямоугольная матрица $\|a_{n*m}\|$. Разработайте программу, определяющую величины:

1) суммы элементов в строках, которые содержат, хотя бы один отрицательный элемент;

2) минимальный элемент матрицы;

3) среднее арифметическое значение четных элементов строк и столбцов матрицы.

Средние арифметические значения четных элементов строк и столбцов матрицы отсортируйте по убыванию методом вставки.

Вариант 7. Дана целочисленная прямоугольная матрица $\|a_{n*m}\|$. Разработайте программу, определяющую величины:

1) количество отрицательных элементов матрицы;

2) максимальный элемент матрицы;

3) суммы элементов матрицы, находящихся ниже главной диагонали.

Элементы матрицы, находящиеся выше главной диагонали, отсортируйте по возрастанию методом обмена.

Вариант 8. Дана целочисленная прямоугольная матрица $\|a_{n*m}\|$. Разработайте программу, определяющую величины:

1) количество положительных элементов матрицы;

2) максимальный и минимальный нечетные элементы строк матрицы;

3) суммы элементов матрицы, находящихся ниже побочной диагонали.

Элементы матрицы, находящиеся ниже побочной диагонали, отсортируйте по возрастанию методом выбора.

Вариант 9. Дана целочисленная прямоугольная матрица $\|a_{n*m}\|$. Разработайте программу, определяющую величины:

1) количество нулевых элементов матрицы;

2) максимальный и минимальный нечетные элементы столбцов матрицы;

3) суммы элементов матрицы, находящихся выше побочной диагонали.

Элементы матрицы, находящиеся выше побочной диагонали, отсортируйте по возрастанию методом вставки.

Вариант 10. Дана целочисленная прямоугольная матрица $\|a_{n*m}\|$. Разработайте программу, определяющую величины:

1) количество ненулевых элементов матрицы;

2) максимальный и минимальный четные элементы столбцов матрицы;

3) суммы элементов матрицы, находящихся ниже побочной диагонали.

Элементы матрицы, находящиеся ниже побочной диагонали, отсортируйте по убыванию методом обмена.

Методические указания

1 Разработайте алгоритм программы.

2 Разработайте программное приложение.

3 Внесите ошибки в программное приложение.

4 Рассчитайте количество ошибок до начала тестирования программы.

5 Проведите тестирование программы.

6 Рассчитайте степень отлаженности программы.

7 Оформите отчет по практической работе, включающий разделы:

- постановка задачи;
- блок-схема программы;
- расчётные данные;
- скриншоты программы;
- код программы;
- выводы.

8 Ответьте на контрольные вопросы.

Контрольные вопросы

1 Какое функциональное назначение модели Миллса?

2 Какие условия учитываются в модели Миллса?

3 Какой вид имеет формула Миллса?

4 Каким образом определяется вероятность наличия первоначальных ошибок?

5 Какие преимущества модели Миллса?

6 Какие недостатки модели Миллса?

4.5 Практическая работа № 8 «Аналитическая статистическая модель надежности. Модель Нельсона»

Цель: закрепить теоретические знания и получить практические навыки в разработке программы оценки показателей надёжности с применением модели Нельсона.

Используемые приемы и технологии: технология визуального проектирования и событийного программирования, среда программирования Visual Studio 2019 Community, язык программирования Visual C++.

Ключевые термины: надежность программного обеспечения, показатели надежности, модель Нельсона.

Варианты заданий

Вариант 1. Дана целочисленная прямоугольная матрица $\|a_{n*m}\|$. Разработайте программу, определяющую величины:

1) количество элементов матрицы, меньших величины B;

2) максимальный и минимальный нечетные элементы столбцов матрицы;

3) суммы элементов матрицы, находящихся выше побочной диагонали.

Элементы матрицы, находящиеся выше побочной диагонали, упорядочите по возрастанию методом вставки.

Вариант 2. Дана целочисленная прямоугольная матрица $\|a_{n*m}\|$. Разработайте программу, определяющую величины:

1) количество элементов матрицы, больших величины B;

2) минимальный по модулю элемент матрицы;

3) суммы элементов, расположенных по периметру матрицы.

Элементы, расположенные по периметру матрицы, упорядочите по убыванию методом обмена.

Вариант 3. Дана целочисленная квадратная матрица $\|a_{n*n}\|$. Разработайте программу, определяющую величины:

1) количество элементов матрицы, находящихся в диапазоне от A до B;

2) максимальный по модулю элемент матрицы;

3) суммы элементов, расположенных на диагоналях матрицы.

Элементы, расположенные на диагоналях матрицы, упорядочите по убыванию методом выбора.

Вариант 4. Дана целочисленная прямоугольная матрица $\|a_{n*m}\|$. Разработайте программу, определяющую величины:

1) количество элементов матрицы, содержащих простые числа;

2) минимум среди модулей элементов, расположенных ниже побочной диагонали;

3) средние квадратичные значения столбцов и строк матрицы.

Средние квадратичные значения столбцов и строк матрицы упорядочите по убыванию методом вставки.

Вариант 5. Дана целочисленная квадратная матрица $\|a_{n*n}\|$. Разработайте программу, определяющую величины:

1) количество элементов матрицы, содержащих простые числа;

2) максимум среди модулей элементов, расположенных выше главной диагонали;

3) средние квадратичные значения четных столбцов и нечетных строк матрицы.

Средние квадратичные значения четных столбцов и нечетных строк матрицы упорядочите по возрастанию методом обмена.

Вариант 6. Дана целочисленная квадратная матрица $\|a_{n*n}\|$. Разработайте программу, определяющую величины:

1) количество элементов матрицы, содержащих простые числа;

2) минимум среди элементов, расположенных выше побочной диагонали;

3) средние квадратичные значения нечетных столбцов и четных строк матрицы.

Средние квадратичные значения нечетных столбцов и четных строк матрицы упорядочите по возрастанию методом выбора.

Вариант 7. Дана целочисленная квадратная матрица $\|a_{n*n}\|$. Разработайте программу, определяющую величины:

1) количество элементов матрицы, содержащих простые числа;

2) максимум среди элементов побочной диагонали;

3) средние арифметические значения столбцов и строк матрицы.

Средние арифметические значения столбцов и строк матрицы упорядочите по возрастанию методом вставки.

Вариант 8. Дана целочисленная квадратная матрица $\|a_{n*n}\|$. Разработайте программу, определяющую величины:

1) количество элементов матрицы, содержащих простые числа;

2) минимум среди элементов главной диагонали;

3) средние арифметические значения четных столбцов и нечетных строк матрицы.

Средние арифметические значения четных столбцов и нечетных строк матрицы упорядочите по убыванию методом обмена.

Вариант 9. Дана целочисленная квадратная матрица $\|a_{n*n}\|$. Разработайте программу, определяющую величины:

1) количество элементов матрицы, содержащих нулевые значения;

2) максимум элементов, расположенных ниже побочной диагонали;

3) суммы положительных элементов столбцов и строк матрицы.

Суммы положительных элементов столбцов и строк матрицы упорядочите по убыванию методом выбора.

Вариант 10. Дана целочисленная квадратная матрица $\|a_n*_n\|$. Разработайте программу, определяющую величины:

1) количество элементов матрицы, содержащих простые числа;

2) минимум элементов, расположенных выше побочной диагонали;

3) суммы отрицательных элементов столбцов и строк матрицы.

Суммы отрицательных элементов столбцов и строк матрицы упорядочите по убыванию методом вставки.

Методические указания

1 Разработайте алгоритм программы.

2 Разработайте программное приложение.

3 Рассчитайте оценки надёжности программы.

4 Оформите отчет по практической работе, включающий разделы:

- постановка задачи;
- блок-схема программы;
- расчётные данные;
- скриншоты программы;
- код программы;
- выводы.

5 Ответьте на контрольные вопросы.

Контрольные вопросы

1 Какое функциональное назначение модели Нельсона?

2 Какие ограничения накладываются на модель Нельсона?

3 По какой формуле оценивается надёжность программного обеспечения?

4 Какие преимущества модели Нельсона?

5 Какие недостатки модели Нельсона?

5 Тестирование программных средств

5.1 Критерии структурного тестирования

Структурное тестирование – процесс обнаружения ошибок в логике программы. Методы структурного тестирования:

- покрытие операторов;
- покрытие решений;
- покрытие условий;
- покрытие решений/условий;
- комбинаторное покрытие условий.

Покрытие кода тестами – мера, показывающая, на сколько процентов исходный код программы был протестирован.

Структурные критерии используют модель программы в виде «белого ящика». Структурные критерии базируются на основных элементах управляющего потокового графа, операторах, ветвях и путях:

- критерий тестирования команд (критерий С0) – набор тестов, обеспечивающий прохождение команды не менее одного раза;
- критерий тестирования ветвей (критерий С1) – набор тестов, обеспечивающий прохождение каждой ветви не менее одного раза.
- критерий тестирования путей (критерий С2) – набор тестов, обеспечивающий прохождение каждого пути не менее одного раза [4–7].

5.2 Практическая работа № 9
«Структурное тестирование программ»

Цель: закрепить теоретические знания и получить практические навыки в разработке программы тестирования методами по стратегии «белого ящика».

Используемые приемы и технологии: технология визуального проектирования и событийного программирования, среда программирования Visual Studio 2019 Community, язык программирования Visual C++.

Ключевые термины: логика программы, критерии структурного тестирования, покрытие, методы структурного тестирования.

Варианты заданий

Протестируйте методами «белого ящика» программу, вычисляющую функцию, где a, b, c –действительные числа.

Вариант 1

$$y = \begin{cases} a * x^2 + b, при\ x + 2 < 0, \text{b} \neq 0 \\ \dfrac{\text{x - a}}{\text{x - c}}, при\ x + 2 > 0\ и\ c \neq 0 \\ \dfrac{10 * x}{c - 4}, в\ остальных\ случаях \end{cases}$$

Вариант 2

$$y = \begin{cases} \dfrac{1}{a * x} - b, при\ x + 5 < 0, \text{c} = 0 \\ \dfrac{\text{x - a}}{\text{x}}, при\ x + 5 > 0\ и\ c \neq 0 \\ \dfrac{10 * x}{c - 4}, в\ остальных\ случаях \end{cases}$$

Вариант 3

$$y = \begin{cases} a * x^2 + b * x + c, при\ a < 0, \text{c} \neq 0 \\ \dfrac{\text{- a}}{\text{x - c}}, при\ a > 0\ и\ c = 0 \\ a * (x + c), в\ остальных\ случаях \end{cases}$$

Вариант 4

$$y = \begin{cases} -a * x - c, при\ c < 0, \text{x} \neq 0 \\ \dfrac{\text{x - a}}{\text{- c}}, при\ c > 0\ и\ x = 0 \\ \dfrac{b * x}{c - a}, в\ остальных\ случаях \end{cases}$$

Вариант 5

$$y = \begin{cases} a - \dfrac{x}{10 + b}, при\ x < 0, \text{b} \neq 0 \\ \dfrac{\text{x - a}}{\text{x - c}}, при\ x > 0\ и\ b = 0 \\ 3 * x + \dfrac{2}{c}, в\ остальных\ случаях \end{cases}$$

Вариант 6

$$y = \begin{cases} a * x^2 + b^2 * x, при\ c < 0, \text{b} \neq 0 \\ \dfrac{\text{x + a}}{\text{x + c}}, при\ c > 0\ и\ b = 0 \\ \dfrac{x}{c}, в\ остальных\ случаях \end{cases}$$

Вариант 7

$$y = \begin{cases} -a * x^2 - b, при\ x < 5, \text{c} \neq 0 \\ \dfrac{\text{x - a}}{\text{x}}, при\ x > 5\ и\ c = 0 \\ \dfrac{-x}{c}, в\ остальных\ случаях \end{cases}$$

Вариант 8

$$y = \begin{cases} -a * x^2, при\ c < 0, \text{a} \neq 0 \\ \dfrac{\text{a - x}}{\text{c * x}}, при\ c > 0\ и\ a = 0 \\ \dfrac{x}{c}, в\ остальных\ случаях \end{cases}$$

Вариант 9

$$y = \begin{cases} a * x^2 + b^2 * x, при\ a < 0, \text{x} \neq 0 \\ x - \dfrac{\text{a}}{\text{x - c}}, при\ a > 0\ и\ x = 0 \\ 1 + \dfrac{x}{c}, в\ остальных\ случаях \end{cases}$$

Вариант 10

$$y = \begin{cases} a * x^2 - b * x + c, при\ x < 3, \text{b} \neq 0 \\ \dfrac{\text{x - a}}{\text{x - c}}, при\ x > 3\ и\ b = 0 \\ \dfrac{x}{c}, в\ остальных\ случаях \end{cases}$$

Методические указания

1 Разработайте алгоритм решения задачи.

2 Разработайте визуальное приложение, осуществляющее тестирование программы по вариантам задания методами структурного тестирования:

- покрытие операторов;
- покрытие решений (покрытие переходов);
- покрытие условий;
- покрытие условий/решений;
- комбинаторное покрытие условий.

3 Результаты тестирования представьте в виде таблиц.

4 Оформите отчет по практической работе, включающий разделы:

- постановка задачи;
- блок-схема программы;
- тестовые варианты и результаты тестирования;
- скриншоты программы;
- код программы (функции обработчиков событий);
- выводы.

5 Ответьте на контрольные вопросы.

Контрольные вопросы

1 Какие этапы включает реализация и тестирование программного продукта?

2 Какие виды тестирования?

3 Какие критерии выбора тестов?

4 Какие свойства тестов?

5 Какие критерии надежности программ?

6 Какова оценка надежности программ?

5.3 Функциональные критерии тестирования

Функциональное тестирование – процесс обнаружения ошибок в функциях программы на всей области определения. Методы функционального тестирования:

- эквивалентное разбиение (классы эквивалентности);
- анализ граничных значений;
- метод функциональных диаграмм;
- предположение об ошибке.

Функциональный критерий обеспечивает контроль степени выполнения требований заказчика в программном продукте. Функциональные критерии подразделяются на виды:

- тестирование проектов спецификации – набор тестов, обеспечивающий проверку каждого тестируемого пункта не менее одного раза;
- тестирование классов входных данных – набор тестов, обеспечивающий проверку представителя каждого класса входных данных не менее одного раза;
- тестирование правил – набор тестов, обеспечивающий проверку каждого правила, если входные и выходные значения описываются набором правил грамматики;
- тестирование классов выходных данных – набор тестов, обеспечивающий проверку представителя каждого выходного класса, при условии, что выходные результаты заранее расклассифицированы, отдельные классы результатов учитывают ограничения на ресурсы или на время (time out);
- тестирование функций – набор тестов, обеспечивающий проверку каждого действия, реализуемого тестируемым модулем, не менее одного раза;
- комбинированные критерии для программ и спецификаций – набор тестов, обеспечивающий проверку всех комбинаций непротиворечивых условий программ и спецификаций не менее одного раза [4–9].

5.4 Практическая работа № 10
«Функциональное тестирование программ»

Цель: закрепить теоретические знания и получить практические навыки в разработке программы тестирования методом функциональных диаграмм.

Используемые приемы и технологии: технология визуального проектирования и событийного программирования, среда программирования Visual Studio 2019 Community, язык программирования Visual C++.

Ключевые термины: диаграмма причинно-следственных связей, функции, ограничения, таблица решений с ограниченным входом, причина, следствие.

Варианты заданий

Вариант 1. Решите биквадратное уравнение $a * x^4 + b * x^2 + c = 0$.

Вариант 2. Решите алгебраическое уравнение 3-й степени (кубическое уравнение) $a * x^3 + b * x^2 + c * x + d = 0$. Корни рассчитайте по тригонометрической формуле Виета.

Вариант 3. Решите алгебраическое уравнение 3-й степени (кубическое уравнение) $a * x^3 + b * x^2 + c * x + d = 0$. Корни приведенного уравнения рассчитайте по формулам Кардано.

Вариант 4. Решите алгебраическое уравнение 2-й степени (квадратное уравнение) $a * x^2 + b * x + c = 0$.

Вариант 5. Дано действительное положительное число ε. Методом касательных вычислите с точностью ε корень уравнения $x^4 - 3 * x^2 + 75 * x - 10000 = 0$, (-11). В круглых скобках указано начальное приближение к корню.

Вариант 6. Дано действительное положительное число ε. Методом деления отрезка пополам найдите приближённое значение корня уравнения $x^3 - 0{,}2 * x^2 - 0{,}2 * x - 1{,}2 = 0$, $[1{,}0;\ 1{,}5]$. Абсолютная погрешность найденного значения не должна превосходить ε. В квадратных скобках указан отрезок, содержащий корень.

Вариант 7. Дано действительное положительное число ε. Методом итераций вычислите с точностью ε корень уравнения $x^3 + 12 * x - 2 = 0$, $(0{,}95)$. В круглых скобках указано начальное приближение к корню.

Вариант 8. Дано действительное положительное число ε. Методом хорд вычислите с точностью ε корень уравнения $\frac{2 * \sin^2 2 * x}{3} - \frac{3 * \cos^2 2 * x}{4} = 0$, $[0;\ \pi/4]$. В квадратных скобках указан отрезок, содержащий корень.

Вариант 9. Дано действительное положительное число ε. Методом касательных вычислите с точностью ε корень уравнения $1{,}8 * x^4 - \sin 10 * x = 0$, $(0{,}22)$. В круглых скобках указано начальное приближение к корню.

Вариант 10. Дано действительное положительное число ε. Методом деления отрезка пополам найдите приближённое значение корня уравнения $x^4 + 2 * x^3 - x - 1 = 0$, $[0;\ 1]$. Абсолютная погрешность найденного значения не должна превосходить ε. В квадратных скобках указан отрезок, содержащий корень.

Методические указания

1 Разработайте алгоритм программы.

2 Разработайте программное приложение, осуществляющее тестирование программы методом диаграмм причинно-следственных связей:

- построение функциональной диаграммы;
- построение таблицы решений;
- определение количества тестовых вариантов;
- генерация тестовых вариантов;
- представление таблиц результатов тестирования.

3 Оформите отчет по практической работе, включающий разделы:

- постановка задачи;
- блок-схема программы;
- разбиение внешней спецификации на отдельные функции, комбинаторные свойства которых должны тестироваться;
- анализ спецификации в поисках условий на входе и действий на выходе;
- построение функциональной диаграммы;
- преобразование функциональной диаграммы в таблицу решений с ограниченным входом;
- таблицы результатов тестирования;
- скриншоты программы;
- код программы (функции обработчиков событий);
- выводы.

Контрольные вопросы

1 Что называется функциональной диаграммой?

2 Для чего предназначены функции диаграммы причинно-следственных связей? Сколько функций? Какие функции?

3 Для чего предназначены ограничения диаграммы причинно-следственных связей? Сколько ограничений? Какие ограничения?

4 Для чего создается таблица решений с ограниченным входом?

5 Какие этапы метода функциональных диаграмм?

6 Какие преимущества и недостатки метода причинно-следственных диаграмм?

5.5 Методы интеграционного тестирования

Интеграционное тестирование – тестирование программного обеспечения на этапе сборки модулей в единый комплекс. Цель интеграционного тестирования – нахождение проблем взаимодействия модулей (компонент, подсистем). Тестирование выполняется через интерфейс модулей с использованием метода «чёрного ящика» [5–9].

Существуют два метода сборки модулей:

- монолитный, характеризующийся одновременным объединением модулей в тестируемый комплекс;
- инкрементальный, характеризующийся пошаговым (помодульным) наращиванием комплекса программ с пошаговым тестированием собираемого комплекса [4].

В инкрементальном методе выделяют две стратегии добавления модулей:

- восходящее тестирование;
- нисходящее тестирование.

При восходящем тестировании сначала тестируются терминальные модули, независящие от других модулей, затем тестируются модули, которые зависят от проверенных модулей. Для модуля разрабатывают драйвер.

Драйвер – программный модуль, эмулирующий окружение модуля при тестировании. Драйвер вызывает тестируемый модуль, передает тестовые данные, проверяет результаты работы модуля и корректность реализации его интерфейса. При восходящем тестировании упрощается локализация ошибок.

При нисходящем тестировании сначала тестируют верхний управляющий модуль программной системы без модулей низкого уровня, вместо которых используют заглушки, затем тестируют модули более низкого уровня.

Заглушка – программный модуль, обладающий тем же интерфейсом, что и замещаемый модуль нижележащего уровня. Заглушка не реализует функциональность замещаемого модуля, а возвращает значения, позволяющие проверить функционирование тестируемого модуля [6].

5.6 Практическая работа № 11 «Интеграционное тестирование программ»

Цель: закрепить теоретические знания и получить практические навыки в разработке программы тестирования методом нисходящего тестирования.

Используемые приемы и технологии: среда программирования Visual Studio 2019 Community, язык программирования Visual C++.

Ключевые термины: интеграционное тестирование, модуль-заглушка, категории заглушек, монолитная сборка модулей, пошаговая сборка модулей.

Варианты заданий

Вариант 1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислите:

1) сумму отрицательных элементов массива;

2) произведение элементов массива, расположенных между максимальным и минимальным элементами.

Упорядочите элементы массива по возрастанию.

Вариант 2. В одномерном массиве, состоящем из *n* вещественных элементов, вычислите:

1) сумму положительных элементов массива;

2) произведение элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементами.

Упорядочите элементы массива по убыванию.

Вариант 3. В одномерном массиве, состоящем из *n* целых элементов, вычислите:

1) произведение элементов массива с четными номерами;

2) сумму элементов массива, расположенных между первым и последним нулевыми элементами.

Преобразуйте массив таким образом, чтобы сначала располагались все положительные элементы, а потом – все отрицательные (элементы, равные нулю, считать положительными).

Вариант 4. В одномерном массиве, состоящем из *n* вещественных элементов, вычислите:

1) сумму элементов массива с нечетными номерами;

2) сумму элементов массива, расположенных между первым и последним отрицательными элементами.

Преобразуйте массив, удалив из него все элементы, модуль которых не превышает 1. Освободившиеся в конце массива элементы заполните нулями.

Вариант 5. В одномерном массиве, состоящем из *n* вещественных элементов, вычислите:

1) максимальный элемент массива;

2) сумму элементов массива, расположенных до последнего положительного элемента.

Преобразуйте массив, удалив из него все элементы, модуль которых находится в интервале [a, b]. Освободившиеся в конце массива элементы заполните нулями.

Вариант 6. В одномерном массиве, состоящем из *n* вещественных элементов, вычислите:

1) минимальный элемент массива;

2) сумму элементов массива, расположенных между первым и последним положительными элементами.

Преобразуйте массив таким образом, чтобы сначала располагались все элементы, равные нулю, а потом – все остальные.

Вариант 7. В одномерном массиве, состоящем из *n* целых элементов, вычислите:

1) номер максимального элемента массива;

2) произведение элементов массива, расположенных между первым и вторым нулевыми элементами.

Преобразуйте массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в нечетных позициях, а во второй половине массива – элементы, стоявшие в четных позициях.

Вариант 8. В одномерном массиве, состоящем из *n* вещественных элементов, вычислите:

1) номер минимального элемента массива;

2) сумму элементов массива, расположенных между первым и вторым отрицательными элементами.

Преобразуйте массив таким образом, чтобы сначала располагались все элементы, модуль которых не превышает 1, а потом – все остальные.

Вариант 9. В одномерном массиве, состоящем из *n* вещественных элементов, вычислите:

1) максимальный по модулю элемент массива;

2) сумму элементов массива, расположенных между первым и вторым положительными элементами.

Преобразуйте массив таким образом, чтобы элементы, равные нулю, располагались после всех остальных.

Вариант 10. В одномерном массиве, состоящем из *n* целых элементов, вычислите:

1) минимальный по модулю элемент массива;

2) сумму модулей элементов массива, расположенных после первого элемента, равного нулю.

Преобразуйте массив таким образом, чтобы в первой половине массива располагались элементы, стоявшие в четных позициях, а во второй половине массива – элементы, стоявшие в нечетных позициях.

Методические указания

1 Разработайте алгоритм программы.

2 Разработайте приложение, осуществляющее тестирование программы методом нисходящего тестирования:

- построение модулей-заглушек;
- тестирование верхнего головного модуля программы;
- тестирование модулей программы;
- представление результатов тестирования.

3 Оформите отчет по практической работе, включающий разделы:

- постановка задачи;
- блок-схема программы;
- структурная схема комплекса программы;

- таблицы результатов тестирования;
- скриншоты программы;
- код программы;
- выводы.

4 Ответьте на контрольные вопросы.

Контрольные вопросы

1 Что называется интеграционным тестированием?
2 Что такое модуль-заглушка?
3 Какие виды модулей-заглушек?
4 Какие этапы алгоритма метода нисходящего тестирования?
5 Какие преимущества и недостатки метода нисходящего тестирования?

6 Стандартизация программного обеспечения

6.1 Стандарты качества программных средств

Качество программных средств поддерживается международными и национальными стандартами. Стандарты качества программных средств подразделяются на четыре группы:

- общие;
- формализующие показатели качества программ;
- отражающие планирование, методы и технологию управления качеством программ;
- поддерживающие технологический процесс создания сложных программных средств высокого качества.

6.2 Уровни показателей качества программных средств

Оценка качества программного средства проводится на стадиях жизненного цикла и включает набор показателей, оценку и сопоставление значений показателей.

Показатели качества подразделяются на четыре уровня:

- факторы качества;
- критерии качества;
- метрики качества;
- оценочные элементы.

Факторы качества – надёжность программных средств, сопровождаемость, удобство применения, эффективность, универсальность, корректность.

Критерии качества – устойчивость функционирования, работоспособность, структурность, простота конструкции, наглядность, повторяемость, лёгкость освоения, доступность эксплуатационных программных документов, удобство эксплуатации и обслуживания, уровень автоматизации, временная

эффективность, ресурсоёмкость, гибкость, мобильность, модифицируемость, полнота реализации, согласованность, логическая корректность, проверенность.

Метрики качества – показатели качества, представляющие абсолютную меру количественной оценки заданного критерия. Метрики составляются из оценочных элементов, определяющих заданное в метрике свойство.

Оценочные элементы – единичные показатели качества программных средств, входящие в метрику качества [10].

6.3 Расчёт показателей качества программных средств

В процессе оценки качества определяются количественные значения абсолютных P_{ij} (j – порядковый номер показателя, i – номер показателя вышестоящего уровня), относительных K_{ij} показателей и базового значения $P_{ij}^{баз}$.

Критерии и метрики характеризуются двумя параметрами:

- количественное значение;
- весовой коэффициент V_{ij}.

Сумма весовых коэффициентов равна 1.

Общая оценка качества определяется экспертами по набору значений оценок факторов качества.

Усреднённое значение оценочного элемента определяется по формуле

$$m_{kq} = \sum_{e=1}^{t} \frac{m_{kqe}}{t}, \qquad (6.1)$$

где e – номер эксперта, $e = \overline{1,t}$;

t – количество экспертов в группе;

q – порядковый номер оценочного элемента, $q = \overline{1,Q}$;

k – порядковый номер метрики.

Итоговая оценка k-ой метрики j-ого критерия определяется по формуле

$$P_{jk}^{M} = \frac{\sum_{q=1}^{Q} m_{kq}}{Q}, \qquad (6.2)$$

где Q – количество оценочных элементов k-ой метрики;

j – порядковый номер критерия качества.

Абсолютное значение j-ого критерия i-ого фактора качества определяется по формуле

$$P_{ij} = \sum_{k=1}^{n} (P_{jk}^{M} * V_{jk}^{M}), \qquad (6.3)$$

где V_{jk}^{M} – весовой коэффициент k-ой метрики j-ого критерия, $k = \overline{1,n}$;

n – число метрик, относящихся к j-ому критерию.

Относительный показатель j-ого критерия i-ого фактора качества определяется по формуле

$$K_{ij} = \frac{P_{ij}}{P_{ij}^{баз}}, \qquad (6.4)$$

где $P_{ij}^{баз}$ – базовое значение j-ого критерия i-ого фактора качества.

Фактор качества определяется по формуле

$$K_{i}^{ф} = \sum_{j=1}^{N} (K_{ij} * V_{ij}^{k}), \qquad (6.5)$$

где V_{ij}^{k} – весовой коэффициент j-ого критерия i-ого фактора качества, $j = \overline{1,N}$;

N – число критериев i-ого фактора качества.

Качество программного средства определяется сравнением расчётных значений показателей с базовыми значениями показателей существующего аналога или расчётного программного средства принимаемого за эталонный образец [10].

6.4 Практическая работа № 12 «Оценка качества программного обеспечения на стадиях жизненного цикла»

Цель: закрепить теоретические знания и получить практические навыки в разработке программы оценки качества программного обеспечения в соответствии с требованиями стандарта ГОСТ 28195-89 «Оценка качества программных средств. Общие положения».

Используемые приемы и технологии: технология визуального проектирования и событийного программирования, среда программирования Visual Studio 2019 Community, язык программирования Visual C++.

Ключевые термины: качество, жизненный цикл, фактор качества, критерий качества, метрика, оценочный элемент, весовой коэффициент, экспертный метод.

Варианты заданий

Разработайте визуальное приложение на языке Visual C++, формализующее работу с базой данных Microsoft Access и позволяющее выполнить ввод, вывод, редактирование, сохранение данных и оценку качества разработанного приложения. Установите значения базовых показателей качества и значения весовых коэффициентов метрик и критериев качества. Значения весовых коэффициентов метрик и критериев качества удовлетворяют условиям:

- диапазон значений от нуля до единицы включительно;
- сумма значений равна единице.

Вариант 1. Структура базы данных «Биржа труда» приводится в таблице 6.1. Оцените качество программы на основе факторов «надёжность» и «сопровождаемость» стадии анализа.

Вариант 2. Структура базы данных «Институт селекции растений» приводится в таблице 6.2. Оцените качество программы на основе факторов «сопровождаемость» и «универсальность» стадии проектирования.

Вариант 3. Структура базы данных «Продажа авиабилетов» приводится в таблице 6.3. Оцените качество программы на основе факторов «надёжность» и «корректность» стадии реализации.

Таблица 6.1 – Биржа труда

Поле	Тип	Размер	Описание
cid	Целое, автоувеличение	-	Идентификатор
Number	Целое	50	Номер безработного
Fio	Текстовый	50	ФИО
Pol	Текстовый	1	Пол
Age	Дата/время	-	Дата рождения
Address	Текстовый	50	Адрес
Phone	Текстовый	50	Телефон

Таблица 6.2 – Институт селекций растений

Поле	Тип	Размер	Описание
cid	Целое, автоувеличение	-	Идентификатор
Name	Текстовый	50	Название сорта
Data	Дата/Время	-	Дата выведения сорта
Svo	Текстовый	50	Характеристика сорта
Address	Текстовый	50	Адрес
Phone	Текстовый	50	Телефон
Email	Текстовый	50	Электронный адрес

Вариант 4. Структура базы данных «Судоходная компания» приводится в таблице 6.4. Оцените качество программы на основе факторов «надёжность» и «эффективность» стадии тестирования.

Таблица 6.3 – Продажа авиабилетов

Поле	Тип	Размер	Описание
cid	Целое, автоувеличение	-	Идентификатор
Num	Тестовый	10	Номер рейса
City	Текстовый	50	Пункт прибытия
Data	Дата/Время	-	Дата вылета
Fio	Тестовый	50	ФИО пассажира
Passport	Текстовый	50	Паспорт
Phone	Текстовый	50	Телефон

Вариант 5. Структура базы данных «Ремонт видеоаппаратуры» приводится в таблице 6.5. Оцените качество программы на основе факторов «удобство применения» и «сопровождаемость» стадии изготовления.

Вариант 6. Структура базы данных «Автосервис» приводится в таблице 6.6. Оцените качество программы на основе факторов «универсальность» и «корректность» стадии обслуживания.

Таблица 6.4 – Судоходная компания

Поле	Тип	Размер	Описание
cid	Целое, автоувеличение	-	Идентификатор
Number	Текстовый	50	Регистрационный номер судна
Name	Текстовый	50	Название судна
Data	Дата/время	-	Дата постройки
Port	Текстовый	50	Порт приписки
Ton	Числовой	10	Водоизмещение
Img	Текстовый	50	Фотография судна

Таблица 6.5 – Ремонт видеоаппаратуры

Поле	Тип	Размер	Описание
cid	Целое, автоувеличение	-	Идентификатор
Card	Числовой	10	Номер квитанции
Fio	Текстовый	50	ФИО заказчика
Data	Дата/время	-	Дата приёма заказа
Address	Текстовый	50	Адрес
Phone	Текстовый	50	Телефон
Price	Числовой	10	Стоимость

Вариант 7. Структура базы данных «Торговая компания» приводится в таблице 6.7. Оцените качество программы на основе факторов «сопровождаемость» и «корректность» стадии тестирования.

Таблица 6.6 – Автосервис

Поле	Тип	Размер	Описание
cid	Целое, автоувеличение	-	Идентификатор
Nazv	Текстовый	50	Название предприятия
Nomer	Числовой	20	ИНН предприятия
Quant	Числовой	10	Количество стояночных мест
Address	Текстовый	50	Адрес
Phone	Текстовый	50	Телефон
Email	Текстовый	50	Электронный адрес

Вариант 8. Структура базы данных «Бюро технической инвентаризации» приводится в таблице 6.8. Оцените качество программы на основе факторов «сопровождаемость» и «эффективность» стадии реализации.

Таблица 6.7 – Торговая компания

Поле	Тип	Размер	Описание
cid	Целое, автоувеличение	-	Идентификатор
Name	Текстовый	50	Название
Account	Числовой	10	Номер счёта в банке
Data	Data/time	-	Дата регистрации
Product	Текстовый	50	Товар
Address	Текстовый	50	Адрес
Phone	Текстовый	50	Телефон
Email	Текстовый	50	Электронный адрес

Таблица 6.8 – Бюро технической инвентаризации

Поле	Тип	Размер	Описание
cid	Целое, автоувеличение	-	Идентификатор
Number	Числовой	10	Номер домовладения
Address	Текстовый	50	Адрес
Data	Дата/время	-	Дата инвентаризации
Squar	Числовой	10	Площадь застройки
Phone	Тестовый	50	Телефон
Email	Текстовый	50	Электронный адрес

Вариант 9. Структура базы данных «Отдел кадров» приводится в таблице 6.9. Оцените качество программы на основе факторов «удобство применения» и «универсальность» стадии тестирования.

Таблица 6.9 – Отдел кадров

Поле	Тип	Размер	Описание
cid	Целое, автоувеличение	-	Идентификатор
Number	Числовой	10	Номер сотрудника
Fio	Текстовый	50	ФИО сотрудника
Data	Дата/время	-	Дата рождения
Address	Текстовый	50	Адрес
Phone	Текстовый	50	Телефон
Email	Текстовый	50	Электронный адрес

Вариант 10. Структура базы данных «Приёмная комиссия университета» приводится в таблице 6.10.

Таблица 6.10 – Приёмная комиссия университета

Поле	Тип	Размер	Описание
cid	Целое, автоувеличение	-	Идентификатор
Number	Целое	50	Регистрационный номер безработного
Fam	Текстовый	50	ФИО абитуриента
Pol	Текстовый	1	Пол
Age	Дата/время	-	Дата рождения
Address	Текстовый	50	Адрес
Phone	Текстовый	50	Телефон

Оцените качество программы на основе факторов «корректность» и «удобство применения» стадии анализа.

Методические указания

1 Определите критерии, используемые на стадии жизненного цикла.
2 Определите набор метрик критериев качества.
3 Определите набор оценочных элементов метрик качества.
4 Определите численные значения оценочных элементов.
5 Разработайте алгоритм задачи.
6 Разработайте программное приложение, позволяющее рассчитать:

- усреднённые значения оценочных элементов;
- итоговые значения метрик на основе исходных значений оценок элементов;
- абсолютные значения критериев качества на основе вычисленных значений метрик и заданных величин весовых коэффициентов метрик;
- относительное значение критерия качества;
- значение фактора качества по вычисленным относительным значениям качества и заданным значениям весовых коэффициентов;

7 Оформите отчет по практической работе, включающий разделы:

- постановка задачи;
- блок-схема программы;
- расчётные показатели качества программы;
- скриншоты программы;
- код программы;
- выводы.

8 Ответьте на контрольные вопросы.

Контрольные вопросы

1 Какие этапы включает процесс оценки качества программного обеспечения?

2 Какие стадии содержатся в жизненном цикле программного обеспечения?

3 Какие показатели относятся к показателям качества первого уровня?

4 Какие показатели включает фактор качества программы?

5 Что называется критерием качества программы?

6 Что называется метрикой качества программы?

7 Как рассчитываются оценки качества программы первого, второго, третьего и четвёртого уровней?

7 Сертификация программного обеспечения

7.1 Критерии оценки качества

В качестве оценки качества используется своевременность представления запрашиваемой информации.

Критерии оценки качества:

- своевременное представление информации, если среднее время обработки запроса не более заданного времени;
- своевременное представление информации, если вероятность представления выходного документа не менее заранее установленной вероятности.

7.2 Модели обслуживания запросов

Модели обслуживания подразделяются на три вида:

- модели бесприоритетного обслуживания;
- модели обслуживания с относительными приоритетами;
- модели обслуживания с абсолютными приоритетами.

Модель бесприоритетного обслуживания – модель обслуживания запросов в порядке их поступления без учёта приоритета и прерывания обработки запроса.

Модель обслуживания с относительными приоритетами – модель первоочередного обслуживания запросов с наивысшим приоритетом без прерывания выполнения начавшегося запроса.

Модель обслуживания с абсолютными приоритетами – модель первоочередного обслуживания запросов с наивысшим приоритетом с прерыванием выполнения начавшегося запроса.

7.3 Практическая работа № 13 «Оценка качества информационной системы»

Цель: закрепить теоретические знания и получить практические навыки в разработке программы оценки вероятностно-временных характеристик функционирования комплекса программно-технических средств.

Используемые приемы и технологии: технология визуального проектирования и событийного программирования, среда программирования Visual Studio 2019 Community, язык программирования Visual C++.

Ключевые термины: сертификация, запрос, приоритет, момент поступления, время обслуживания своевременность представления информации, среднее время обработки запроса, вероятность представления документа.

Варианты заданий. Модель бесприоритетного обслуживания

Разработайте визуальное приложение на языке Visual C++, формализующее оценку своевременности представления запрашиваемой выходной информации с использованием модели бесприоритетного обслуживания и позволяющее выполнить ввод, расчёт, вывод данных и построение диаграммы поступления и исполнения запросов.

Вариант 1. Запрашивается один документ (таблица 7.1).

Таблица 7.1 – Данные варианта 1

Временные параметры запроса, мин	Тип запроса информации			
	A	B	C	D
Момент поступления	0	2	3	5
Время обработки	3	2	4	1
Среднее время обработки запроса	6,25			

Определите своевременность представления запрашиваемой информации. Измените время обработки на обратный порядок.

Вариант 2. Запрашивается один документ (таблица 7.2).

Таблица 7.2 – Данные варианта 2

Временные параметры запроса, мин	Тип запроса информации			
	A	B	C	D
Момент поступления	0	2	4	6
Время обработки, мин	2	4	3	5
Среднее время обработки запроса	7,55			

Определите своевременность представления запрашиваемой информации. Измените время обработки на обратный порядок.

Вариант 3. Запрашивается один документ (таблица 7.3).

Таблица 7.3 – Данные варианта 3

Временные параметры запроса, мин	Тип запроса информации			
	A	B	C	D
Момент поступления	0	3	6	9
Время обработки, мин	4	5	6	8
Среднее время обработки запроса	12,35			

Определите своевременность представления запрашиваемой информации. Измените время обработки на обратный порядок.

Вариант 4. Запрашивается один документ (таблица 7.4).

Таблица 7.4 – Данные варианта 4

Временные параметры запроса, мин	Тип запроса информации			
	A	B	C	D
Момент поступления	0	3	4	6
Время обработки, мин	2	4	3	5
Среднее время обработки запроса	8,75			

Определите своевременность представления запрашиваемой информации. Измените время обработки на обратный порядок.

Вариант 5. Запрашивается один документ (таблица 7.5).

Таблица 7.5 – Данные варианта 5

Временные параметры запроса, мин	Тип запроса информации			
	A	B	C	D
Момент поступления	0	4	5	7
Время обработки, мин	3	5	7	4
Среднее время обработки запроса	12,15			

Определите своевременность представления запрашиваемой информации. Измените время обработки на обратный порядок.

Вариант 6. Запрашивается один документ (таблица 7.6). Определите своевременность представления запрашиваемой информации. Измените время обработки на обратный порядок.

Вариант 7. Запрашивается один документ (таблица 7.7). Определите своевременность представления запрашиваемой информации. Измените время обработки на обратный порядок.

Таблица 7.6 – Данные варианта 6

Временные параметры запроса, мин	Тип запроса информации			
	A	B	C	D
Момент поступления	0	3	5	8
Время обработки, мин	4	6	3	7
Среднее время обработки запроса	11,45			

Таблица 7.7 – Данные варианта 7

Временные параметры запроса, мин	Тип запроса информации			
	A	B	C	D
Момент поступления	0	2	4	8
Время обработки, мин	26	5	8	4
Среднее время обработки запроса	14,45			

Вариант 8. Запрашивается один документ (таблица 7.8).

Таблица 7.8 – Данные варианта 8

Временные параметры запроса, мин	Тип запроса информации			
	A	B	C	D
Момент поступления	0	4	8	12
Время обработки, мин	6	8	7	9
Среднее время обработки запроса	17,25			

Определите своевременность представления запрашиваемой информации. Измените время обработки на обратный порядок.

Вариант 9. Запрашивается один документ (таблица 7.9).

Таблица 7.9 – Данные варианта 9

Временные параметры запроса, мин	Тип запроса информации			
	A	B	C	D
Момент поступления	0	2	5	6
Время обработки, мин	5	8	6	4
Среднее время обработки запроса	14,75			

Определите своевременность представления запрашиваемой информации. Измените время обработки на обратный порядок.

Вариант 10. Запрашивается один документ (таблица 7.10). Определите своевременность представления запрашиваемой информации. Измените время обработки на обратный порядок.

Таблица 7.10 – Данные варианта 10

Временные параметры запроса, мин	Тип запроса информации			
	A	B	C	D
Момент поступления	0	5	8	9
Время обработки, мин	4	6	9	8
Среднее время обработки запроса	4,85			

Варианты заданий. Модель обслуживания с относительными приоритетами

Разработайте визуальное приложение на языке Visual C++, формализующее оценку своевременности представления запрашиваемой выходной информации с использованием модели обслуживания с относительными приоритетами и позволяющее выполнить ввод, расчёт, вывод данных и построение диаграммы поступления и исполнения запросов.

Вариант 1. Запрашивается один документ (таблица 7.11).

Таблица 7.11 – Данные варианта 1

Параметры запроса информации	Тип запроса информации				
	A	B	C	D	E
Приоритет	4	1	3	2	5
Момент поступления, мин	0	1	3	5	4
Время обработки, мин	3	5	2	6	3
Среднее время обработки запроса, мин	12,15				

Определите своевременность представления запрашиваемой информации.

Вариант 2. Запрашивается один документ (таблица 7.12).

Таблица 7.12 – Данные варианта 2

Параметры запроса информации	Тип запроса информации				
	A	B	C	D	E
Приоритет	3	1	2	5	5
Момент поступления, мин	0	3	2	5	7
Время обработки, мин	2	4	3	7	5
Среднее время обработки запроса, мин	10,65				

Определите своевременность представления запрашиваемой информации.

Вариант 3. Запрашивается один документ (таблица 7.13). Определите своевременность представления запрашиваемой информации.

Таблица 7.13 – Данные варианта 3

Параметры запроса информации	Тип запроса информации				
	A	B	C	D	E
Приоритет	3	2	4	5	1
Момент поступления, мин	0	2	4	5	6
Время обработки, мин	4	5	3	6	8
Среднее время обработки запроса, мин	15,25				

Вариант 4. Запрашивается один документ (таблица 7.14).

Таблица 7.14 – Данные варианта 4

Параметры запроса информации	Тип запроса информации				
	A	B	C	D	E
Приоритет	2	1	5	3	4
Момент поступления, мин	0	2	4	6	8
Время обработки, мин	3	4	4	5	3
Среднее время обработки запроса, мин	11,15				

Определите своевременность представления запрашиваемой информации.

Вариант 5. Запрашивается один документ (таблица 7.15).

Таблица 7.15 – Данные варианта 5

Параметры запроса информации	Тип запроса информации				
	A	B	C	D	E
Приоритет	3	2	1	5	4
Момент поступления, мин	0	4	5	7	9
Время обработки, мин	2	3	4	4	6
Среднее время обработки запроса, мин	14,75				

Определите своевременность представления запрашиваемой информации.

Вариант 6. Запрашивается один документ (таблица 7.16). Определите своевременность представления запрашиваемой информации.

Вариант 7. Запрашивается один документ (таблица 7.17). Определите своевременность представления запрашиваемой информации.

Вариант 8. Запрашивается один документ (таблица 7.18). Определите своевременность представления запрашиваемой информации.

Вариант 9. Запрашивается один документ (таблица 7.19). Определите своевременность представления запрашиваемой информации.

Вариант 10. Запрашивается один документ (таблица 7.20). Определите своевременность представления запрашиваемой информации.

Таблица 7.16 – Данные варианта 6

Параметры запроса информации	Тип запроса информации				
	A	B	C	D	E
Приоритет	3	4	5	1	2
Момент поступления, мин	0	2	6	5	4
Время обработки, мин	1	4	3	2	6
Среднее время обработки запроса, мин	9,15				

Таблица 7.17 – Данные варианта 7

Параметры запроса информации	Тип запроса информации				
	A	B	C	D	E
Приоритет	2	4	3	4	5
Момент поступления, мин	0	4	5	7	8
Время обработки, мин	3	2	6	5	4
Среднее время обработки запроса, мин	11,85				

Таблица 7.18 – Данные варианта 8

Параметры запроса информации	Тип запроса информации				
	A	B	C	D	E
Приоритет	5	2	4	1	3
Момент поступления, мин	0	3	4	6	8
Время обработки, мин	4	4	2	3	5
Среднее время обработки запроса, мин	11,40				

Таблица 7.19 – Данные варианта 9

Параметры запроса информации	Тип запроса информации				
	A	B	C	D	E
Приоритет	3	2	1	5	4
Момент поступления, мин	0	2	6	7	8
Время обработки, мин	3	4	2	1	5
Среднее время обработки запроса, мин	9,65				

Варианты заданий. Модель обслуживания с абсолютными приоритетами

Разработайте визуальное приложение на языке Visual C++, формализующее оценку своевременности представления запрашиваемой выходной информации с использованием модели обслуживания с абсолютными приоритетами, и позволяющее выполнить ввод, расчёт, вывод данных и построение диаграммы поступления и исполнения запросов.

Таблица 7.20 – Данные варианта 10

Параметры запроса информации	Тип запроса информации				
	A	B	C	D	E
Приоритет	5	2	1	3	4
Момент поступления, мин	0	3	4	6	7
Время обработки, мин	2	5	4	2	1
Среднее время обработки запроса, мин	10,25				

Вариант 1. Запрашивается один документ (таблица 7.21).

Таблица 7.21 – Данные варианта 1

Параметры запроса информации	Тип запроса информации				
	A	B	C	D	E
Приоритет	2	1	5	3	4
Момент поступления, мин	0	1	2	3	4
Время обработки, мин	3	4	2	5	3
Среднее время обработки запроса, мин	10,25				

Определите своевременность представления запрашиваемой информации.
Вариант 2. Запрашивается один документ (таблица 7.22).

Таблица 7.22 – Данные варианта 2

Параметры запроса информации	Тип запроса информации				
	A	B	C	D	E
Приоритет	5	4	3	1	2
Момент поступления, мин	0	3	5	4	6
Время обработки, мин	4	5	3	5	2
Среднее время обработки запроса, мин	14,15				

Определите своевременность представления запрашиваемой информации.
Вариант 3. Запрашивается один документ (таблица 7.23).

Таблица 7.23 – Данные варианта 3

Параметры запроса информации	Тип запроса информации				
	A	B	C	D	E
Приоритет	4	2	1	3	5
Момент поступления, мин	0	2	3	5	7
Время обработки, мин	2	4	5	3	2
Среднее время обработки запроса, мин	11,25				

Определите своевременность представления запрашиваемой информации.
Вариант 4. Запрашивается один документ (таблица 7.24).

Таблица 7.24 – Данные варианта 4

Параметры запроса информации	Тип запроса информации				
	A	B	C	D	E
Приоритет	4	1	3	2	5
Момент поступления, мин	0	3	5	4	2
Время обработки, мин	4	3	4	2	1
Среднее время обработки запроса, мин	10,65				

Определите своевременность представления запрашиваемой информации.

Вариант 5. Запрашивается один документ (таблица 7.25).

Таблица 7.25 – Данные варианта 5

Параметры запроса информации	Тип запроса информации				
	A	B	C	D	E
Приоритет	2	4	5	1	3
Момент поступления, мин	0	4	3	2	4
Время обработки, мин	3	2	4	3	5
Среднее время обработки запроса, мин	10,55				

Определите своевременность представления запрашиваемой информации.

Вариант 6. Запрашивается один документ (таблица 7.26).

Таблица 7.26 – Данные варианта 6

Параметры запроса информации	Тип запроса информации				
	A	B	C	D	E
Приоритет	1	3	2	4	5
Момент поступления, мин	0	4	5	7	6
Время обработки, мин	5	3	2	4	3
Среднее время обработки запроса, мин	10,35				

Определите своевременность представления запрашиваемой информации.

Вариант 7. Запрашивается один документ (таблица 7.27). Определите своевременность представления запрашиваемой информации.

Вариант 8. Запрашивается один документ (таблица 7.28). Определите своевременность представления запрашиваемой информации.

Вариант 9. Запрашивается один документ (таблица 7.29). Определите своевременность представления запрашиваемой информации.

Вариант 10. Запрашивается один документ (таблица 7.30). Определите своевременность представления запрашиваемой информации.

Таблица 7.27 – Данные варианта 7

Параметры запроса информации	Тип запроса информации				
	A	B	C	D	E
Приоритет	1	4	3	5	2
Момент поступления, мин	0	2	4	3	5
Время обработки, мин	3	1	2	4	3
Среднее время обработки запроса, мин	7,25				

Таблица 7.28 – Данные варианта 8

Параметры запроса информации	Тип запроса информации				
	A	B	C	D	E
Приоритет	2	1	5	4	3
Момент поступления, мин	0	4	5	3	2
Время обработки, мин	3	6	4	2	3
Среднее время обработки запроса, мин	11,35				

Таблица 7.29 – Данные варианта 9

Параметры запроса информации	Тип запроса информации				
	A	B	C	D	E
Приоритет	1	3	5	4	2
Момент поступления, мин	0	2	4	3	5
Время обработки, мин	3	1	4	2	3
Среднее время обработки запроса, мин	7,45				

Таблица 7.30 – Данные варианта 10

Параметры запроса информации	Тип запроса информации				
	A	B	C	D	E
Приоритет	3	1	4	5	2
Момент поступления, мин	0	3	5	6	7
Время обработки, мин	4	2	3	2	1
Среднее время обработки запроса, мин	8,95				

Методические указания

1 Разработайте алгоритм задачи.
2 Постройте диаграмму поступления и исполнения запросов.
3 Разработайте программное приложение.
4 Оформите отчет по практической работе, включающий разделы:

- постановка задачи;
- блок-схема программы;
- расчётные данные;
- скриншоты программы;
- код программы;

- выводы.

5 Ответьте на контрольные вопросы.

Контрольные вопросы

1 Какой параметр выбирается в качестве оценки работы программно-технических средств информационной системы?

2 Какие критерии оценки качества информационной системы?

3 В чём особенность модели бесприоритетного обслуживания?

4 В чём особенность модели обслуживания с относительными приоритетами?

5 В чём особенность модели обслуживания с абсолютными приоритетами?

6 Что понимается под диаграммой поступления и исполнения запросов?

ЗАКЛЮЧЕНИЕ

Верификация программного обеспечения позволяет проверить соответствие программного приложения техническому заданию. Цель верификации – обнаружение ошибок, уязвимостей, некорректно реализованных свойств и требований.

В методическом указании приводятся задания к выполнению практических работ по дисциплине «Методы верификации и оценки качества программного обеспечения». Задания сопровождаются кратким теоретическим обоснованием. Выполнение практических работ позволит приобрести навыки оценки качества программных средств и закрепить теоретические знания, полученные студентами на лекциях.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Сандлер, К. Искусство тестирования программ / К. Сандлер, Т. Баджет, Г. Майерс. – Москва : Диалектика, 2019. – 272 с.

2 Ананьева, Т. Н. Стандартизация, сертификация и управление качеством программного обеспечения : учебное пособие / Т. Н. Ананьева, Г. Н. Исаев, Н. Г. Новикова. – Москва : Инфра-М, 2016. – 232 с.

3 Черников, Б. В. Управление качеством программного обеспечения : учебник / Б. В. Черников – Москва : Форум, 2012. – 240 с.

4 Поклонов, Б. Е. Оценка качества программного обеспечения / Б. Е. Поклонов. – Москва : Форум, 2012. – 400 с.

5 Чекал, Е. Г. Надёжность информационных систем : учебное пособие в 2 ч. / Е. Г. Чекал. – Ульяновск : УлГУ, 2012. Ч. 1 – 118 с.

6 Благодатских, В. А. Стандартизация разработки программных средств : учебное пособие / В. А. Благодатских, В. А. Волнин, К. Ф. Поскакалов; под. ред. О. С. Разумова. – Москва : Финансы и статистика, 2005. – 288 с.

7 Крылова, Г. Д. Основы стандартизации, сертификации, метрологии / Г. Д. Крылова. – Москва : ЮНИТИ, 2000. – 215 с.

8 Котляров, В. П. Пинаев Д. В. Оценка степени тестированности программного продукта / В. П. Котляров, Д. В. Пинаев. – Санкт-Петербург, 1997. – 254 с.

9 Ван Тассел, Д. Стиль, разработка, эффективность, отладка и испытание программ / Д. Ван Тассел. – Москва : Мир, 1985. – 280 с.

10 ГОСТ 28195-89. Оценка качества программных средств. Общие положения. – Москва : Государственный комитет СССР по стандартам, 1989. – 254 с.

Семахин Андрей Михайлович

МЕТОДЫ ВЕРИФИКАЦИИ И ОЦЕНКИ КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Методические указания
к выполнению практических работ
для магистрантов направления подготовки 09.04.04
«Программная инженерия»

Редактор Л. П. Чукомина

Подписано в печать 11.03.20	Формат 60x84 1/16	Бумага 80 г/м^2
Печать цифровая	Усл. печ. л. 4,5	Уч.-изд. л. 4,5
Заказ 22	Тираж 25	Не для продажи

БИЦ Курганского государственного университета.
640020, г. Курган, ул. Советская, 63/4.
Курганский государственный университет.