|  |  |  |
| --- | --- | --- |
| Дата | Содержание выполняемой работы | Подпись руководите ля |
| 23.05.2022,  24.05.2022 | **Практическая работа №23-24**  **Тема:** Обоснование выбора средств реализации проекта. Кодирование модулей разрабатываемого программного.  **Наименование работы:** Отладка модулей программной системы. Интегрирование программных модулей.  **Цель:** Закрепление умений и навыков по кодированию, отладке и интегрированию модулей программного средства, корректировке кода готового ПС.  Часть 1.  **Задание 1.**  Изучил теоретический материал и законспектировал основные определения.  Отладка - это некоторый детективный процесс. Программа, в которую внесены изменения, подозревается в том, что она работает некорректно.  Отладка может доказать некорректность программы, но она не может доказать ее правильность. Отладка не гарантирует корректности программы, даже если все тесты пройдены успешно. Искусное тестирование создает высокую степень уверенности в корректности программы.  Локализацией называют процесс определения оператора программы, выполнение которого вызвало нарушение нормального вычислительного процесса. Для исправления ошибки необходимо определить ее причину, т.е. определить оператор или фрагмент, со-держащие ошибку.  Отладка программы в любом случае предполагает обдумывание и логическое осмысление всей имеющейся информации об ошибке. Большинство ошибок можно обнаружить по косвенным признакам посредством тщательного анализа текстов программ и результатов тестирования без получения дополнительной информации.  Метод ручного тестирования - самый простой и естественный способ данной группы. При обнаружении ошибки необходимо выполнить тестируемую программу вручную, используя тестовый набор, при работе с которыми была обнаружена ошибка. Метод очень эффективен, но не применим для больших программ, программ со сложными вычислениями и в тех случаях, когда ошибка связана с неверным представлением программиста о выполнении некоторых операций.  Метод индукции основан на тщательном анализе симптомов ошибки, которые могут проявляться как неверные результаты вычислений или как сообщение об ошибке. Если компьютер просто "зависает", то фрагмент проявления ошибки вычисляют, исходя из последних полученных результатов и действий пользователя. Полученную таким образом информацию организуют и тщательно изучают, просматривая соответствующий фрагмент программы. В результате этих действий выдвигают гипотезы об ошибках, каждую из которых проверяют.  По методу дедукции вначале формируют множество причин, которые могли бы вызвать данное проявление ошибки. Затем анализируя причины, исключают те, которые противоречат имеющимся данным. Если все причины исключены, то следует выполнить дополнительное тестирование исследуемого фрагмента. В противном случае наиболее вероятную гипотезу пытаются доказать. Если гипотеза объясняет полученные признаки ошибки, то ошибка найдена, иначе - проверяют следующую причину.  *Точку останова (*breakpoint) в Visual Studio можно помещать на любую строку кода, которая в действительности выполняется. Самый простой способ — щелчок на необходимой строке в окне редактора кода внутри затененной области вдоль левого края окна документа (или выделение нужной строки и нажатие клавиши <F9>).  После срабатывания точки останова обычно необходимо просмотреть значения переменных. Проще всего это сделать, наведя курсор мыши на имя интересующей переменной прямо в окне редактора кода. Это приводит к появлению небольшого всплывающего окошка, в котором показано значение данной переменной; это окошко можно развернуть, чтобы просмотреть дополнительные детали.  Исключения являются замечательным средством для обеспечения надлежащей обработки ошибок в поставляемом приложении. В случае правильного применения они позволяют обрести уверенность в том, что приложению удастся справиться с трудностями, а перед пользователем никогда не появится диалоговое окно с техническим описанием неполадки.  Компиляция практически всего коммерческого программного обеспечения на стадии отладки и на стадии подготовки окончательной версии продукта должна проводиться немного по-разному. Среда Visual Studio способна понимать это, поскольку сохраняет информацию обо всех параметрах, которые ей надлежит передавать компилятору. Для поддержки разных вариантов компоновки проекта Visual Studio потребуется сохранять подобную информацию в более чем одном экземпляре. Разные экземпляры такой информации называются конфигурациями. При создании проекта Visual Studio автоматически предлагает на выбор две таких конфигурации, которые называются, соответственно, Debug (Отладка) и Release (Выпуск).  **Задание 2.**  Изучил теоретический материал. Законспектировал основные определения.  Исключительная ситуация - это ошибка, которая возникает во время выполнения программы. Используя С# – подсистему обработки исключительных ситуаций, с такими ошибками можно справляться. В С# эта подсистема включает в себя усовершенствованные методы, используемые в языках C++ и Java. Преимущество подсистемы обработки исключений состоит в автоматизации создания большей части кода, который ранее необходимо было вводить в программы "вручную".  Еще одним преимуществом обработки исключительных ситуаций в С# является определение стандартных исключений для таких распространенных программных ошибок, как деление на нуль или попадание вне диапазона определения индекса. Чтобы отреагировать на возникновение таких ошибок, программа должна отслеживать и обрабатывать эти исключения. Без знания возможностей С#-подсистемы обработки исключений успешное программирование на С# попросту невозможно.  Ядром обработки исключений являются блоки try и catch. Эти ключевые слова работают "в одной связке". В С# исключения представляются классами. Все классы исключений должны быть выведены из встроенного класса исключений Exception, который является частью про-странства имен System. Таким образом, все исключения - подклассы класса Exception.  **Задание 3.**  Выполнил отладку кода разрабатываемой системы. Описал обработанные исключительные ситуации в коде программы, листинг кода с обработкой исключений вставил в Приложение 23.1.  Часть 2.  **Задание 1.**  Еще раз проверил совместимость модулей программного средства. При выявлении ошибок исправил их.  **Задание 2.**  Листинг кода и скриншоты окон разработанного программного средства вставил в Приложение 24.1. |  |