**Проект ООТРПО**

**Дайджест по третьему этапу**

**Участники проекта: гр. 5303 Допира В, Бочкарев И, Ильянов В., гр. 5304 Павлов Д**

**Выбранный контейнер: многодольный граф**

**В дайджесте представлены:**

* План итерации.
* Демонстрация работы системы
* Элементы реализации
* Материалы поддержки пользователей
* Компоновка продукта

Система автоматизированной генерации расписания

План итераций

Версия 1.0

**История Ревизий**

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версия** | **Описание** | **Автор** |
| 31.05.2020 | 1.0 | Первоначальная версия | Допира Валерия |

**1. Введение**

**1.1 Цель**

Цель документа – описание плана итерации проекта. В данной итерации проводится подготовка к завершению работ над проектом.

**1.2 Определения и сокращения**

Представлены в артефакте Глоссарий.

**1.3 План**

График сдачи каждой задачи проекта представлен ниже (см. табл.1).

Таблица 1 – График проекта

|  |  |  |  |
| --- | --- | --- | --- |
| ***Фаза*** | ***Задача*** | ***Окончание работы*** | ***Исполнители*** |
| Построение | План итерации | 31.05.2020 | Допира Валерия |
| Демонстрация работы системы | 31.05.2020 | Бочкарев Иван |
| Элементы реализации | 31.05.2020 | Ильянов Вячеслав, Бочкарев Иван,  Павлов Данила |
| Материалы поддержки  пользователей | 31.05.2020 | Ильянов Вячеслав |
| Компоновка продукта | 31.05.2020 | Павлов Данила, Допира Валерия |

**2. Нагрузка исполнителей**

* Допира Валерия

Таблица 2 – Нагрузка на исполнителя 1

|  |  |  |  |
| --- | --- | --- | --- |
| ***Задача*** | ***Окончание работы*** | ***Затраченное время*** | ***Исполнители*** |
| План итерации | 31.05.2020 | 1 день | Допира Валерия |
| Компоновка продукта | 31.05.2020 | 3 дня | Павлов Данила, Допира Валерия |

* Бочкарев Иван

Таблица 3 – Нагрузка на исполнителя 2

|  |  |  |  |
| --- | --- | --- | --- |
| ***Задача*** | ***Окончание работы*** | ***Затраченное время*** | ***Исполнители*** |
| Демонстрация работы системы | 31.05.2020 | 1 день | Бочкарев Иван |
| Элементы реализации | 31.05.2020 | 2 дня | Ильянов Вячеслав, Бочкарев Иван,  Павлов Данила |

* Павлов Данила

Таблица 4 – Нагрузка на исполнителя 3

|  |  |  |  |
| --- | --- | --- | --- |
| ***Задача*** | ***Окончание работы*** | ***Затраченное время*** | ***Исполнители*** |
| Элементы реализации | 31.05.2020 | 2 дня | Ильянов Вячеслав, Бочкарев Иван,  Павлов Данила |
| Компоновка продукта | 31.05.2020 | 3 дня | Павлов Данила, Допира Валерия |

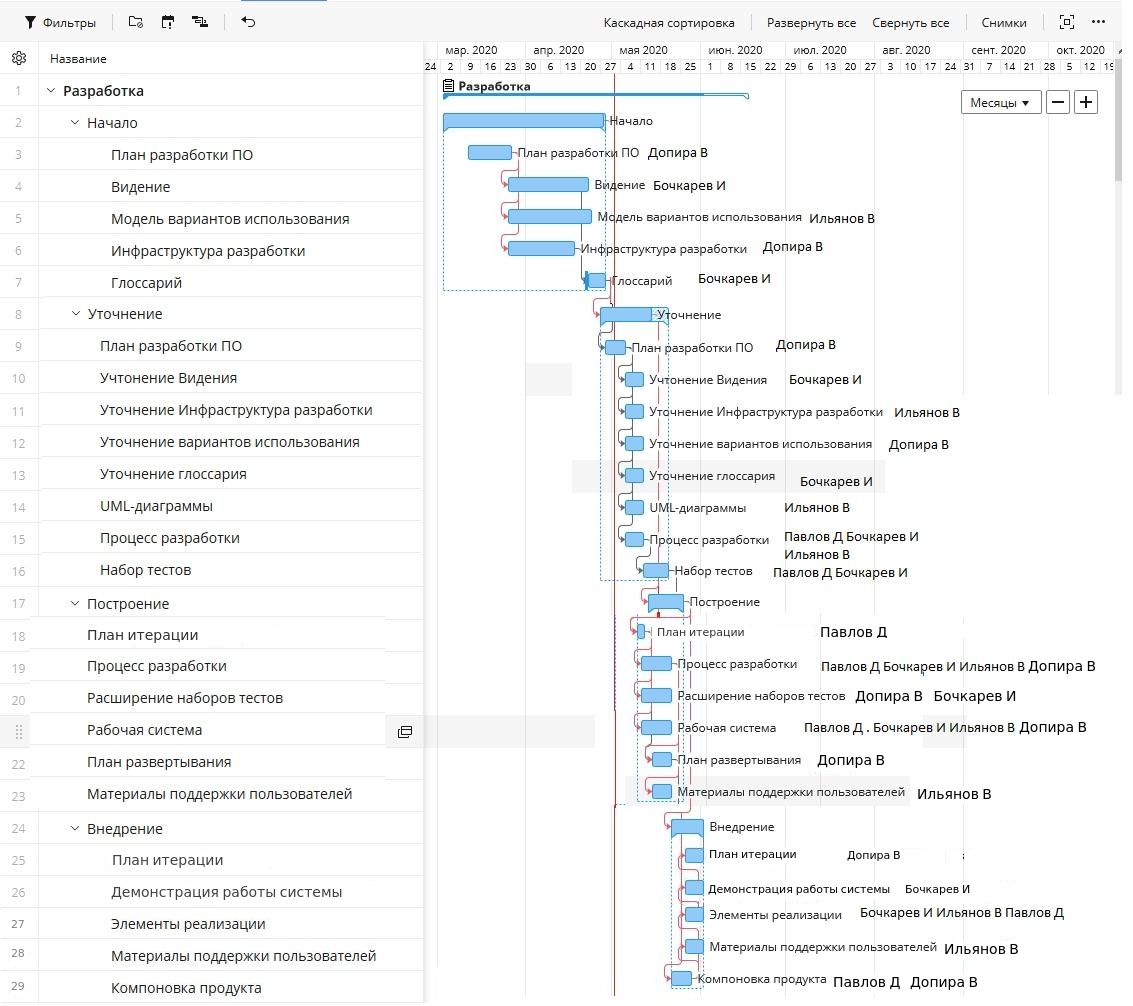
* Ильянов Вячеслав

Таблица 5 – Нагрузка на исполнителя 4

|  |  |  |  |
| --- | --- | --- | --- |
| ***Задача*** | ***Окончание работы*** | ***Затраченное время*** | ***Исполнители*** |
| Элементы реализации | 31.05.2020 | 2 дня | Ильянов Вячеслав, Бочкарев Иван,  Павлов Данила |
| Материалы поддержки  пользователей | 31.05.2020 | 1 день | Ильянов Вячеслав |

**3. Диаграмма Ганта**

Для иллюстрации плана, графика работ и занятости членов команды, работающих над проектом, удобно использовать диаграмму Ганта. Диаграмма Ганта со списком рабочих продуктов и исполнителями представлена ниже (см. рис.1).

Рисунок 1 - Диаграмма Ганта

Система автоматизированной генерации расписания

Демонстрация работы системы

Версия 1.0

**История Ревизий**

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версия** | **Описание** | **Автор** |
| 31.05.2020 | 1.0 | Первоначальная версия | Бочкарев Иван |

**Демонстрация системы**

**Ссылка на видеоматериалы:**

<https://drive.google.com/open?id=15roV6q0P3Y_a0xA8CXMRIs32zrhNJMwD>

**Описание:**

На видео продемонстрирована текущая версия программы, содержащая ключевые компоненты (контейнер). Данные загружаются из файла формата JSON. Также через приложение можно создать новый файл, открыть, изменить и сохранить текущий (сохранить и сохранить как).

При нажатии на элементы таблиц и затем правую мышку открывается контекстное меню. В окне данных их можно добавлять, редактировать и удалять. В окне групп можно соотнести группу, предмет и указать количество часов. Имеются проверки на корректность выбранных данных. Все действия влияют на используемую модель (двудольный граф), добавляя и удаляя ребро между двумя долями. Расписание выводится, если перейти на вкладку «Расписание». Визуализируется граф во вкладке «Визуализация графа», оно пока не до конца доделано. Доступно удаление вершин и фильтрация.

Система автоматизированной генерации расписания

Элементы реализации

Версия 1.0

**История Ревизий**

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версия** | **Описание** | **Автор** |
| 31.05.2020 | 1.0 | Первоначальная версия | Ильянов Вячеслав,  Бочкарев Иван,  Павлов Данила |

**1. Введение**

**1.1. Цель**

Целью данного плана тестирования является описание тестирования разработанного приложения.

**1.2. Контекст**

**1.3. Определения, акронимы, сокращения**

См. глоссарий проекта.

Система автоматизированной генерации расписания

Материалы поддержки пользователей

Версия 1.0

**История Ревизий**

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версия** | **Описание** | **Автор** |
| 31.05.2020 | 1.0 | Первоначальная версия | Ильянов Вячеслав |

**1. Введение**

**1.1 Цель**

Техническая поддержка служит для помощи конкретным пользователям решать возникающие конкретные проблемы с продуктом и его использованием, нежели задачи, связанные с обучением, индивидуальной настройкой или другими услугами поддержки.

**2. Методы поддежки**

Методы поддержки были описаны в 3 итерации. См. Материалы поддержки пользователей.

**3. Техническая поддержка пользователей**

Техническая поддержка (technical support) – это служба, в которую пользователи продукта или услуги могут обратиться за оказанием технической поддержки по решению возникшей проблемы, а также за получением дополнительной информации по интересующему вопросу.

Стоит также отметить, что служба технической поддержки может быть организована и для обслуживания сотрудников компании внутри организации, например, если сотрудникам нужна техническая помощь с компьютерной техникой (сломался компьютер/принтер) или неработающим программным обеспечением.

Отдельно можно выделить техническую поддержку клиентов (или клиентская поддержка). Такой вид поддержки имеет стратегическую направленность и нацелен на выстраивание долгосрочных отношений с клиентами.

## **3.1 Цели технической поддержки:**

* оказание технической помощи пользователям при установке программного продукта
* оказание помощи в освоении и в решении проблем при использовании программного продукта
* сбор и систематизирование замечаний и пожеланий пользователей к программному продукту для улучшения качества использования данного продукта.

**3.2 Обязанности службы:**

* регистрация обращений пользователей в электронной системе
* перенаправление обращений/заявок пользователей к соответствующим специалистам техподдержки для решения проблемы
* ведение журнала с описанием выполненных действий и принятых решений с последующим занесением решений в единую базу
* оказание помощи и консультаций при установке, настройке и обновлении программного продукта, а также по использованию услуги
* оказание помощи в решении технических проблем при использовании продукта/услуги и дальнейшая ее координация
* оказание помощи по восстановлению работоспособности программного продукта после фатальных сбоев
* предоставление технической информации по функциональности продукта/услуги
* анализ проблем и разработка рекомендации по их устранению, которые были выявлены при использовании продукта/услуги
* отвечать на обращение пользователей в установленные сроки
* знание порядка и правил обработки обращений пользователей
* знание стандартных решений и ответов на наиболее часто задаваемые вопросы пользователей
* знание технических характеристик продукта

## **3.3 Виды технической поддержки**

Техподдержку можно разделить на 3 направления:

* поддержка инфраструктуры
* поддержка пользователей
* сопровождение продуктов

Оказание услуг технической поддержки может быть предоставлена как на бесплатной, так и на платной основе. Платная техническая поддержка предоставляется на определенный срок по определенной цене и может включать в себя следующие услуги:

* круглосуточный мониторинг
* круглосуточная техподдержка
* техническая помощь с выездом специалиста т.е. помощь “на месте”
* резервное копирование, аварийное восстановление

Наиболее часто за оказанием технической поддержки можно обратиться через чат, электронную почту, форму “обратной связи”, телефон, панель управления, и, как правило, по рабочим дням.

В случае обращения пользователя, специалисты из техподдержки обязаны выяснить нюансы проблемы, найти и предложить способы решения этой проблемы, а также дать рекомендации по дальнейшему предотвращению подобной проблемы.

Обработка обращений пользователей и клиентов происходит с помощью программного обеспечения, например, Helpdesk и Salesforce.

**3.4 Поддержка разрабатываемого продукта**

Пока у разработчиков продукта нет достаточного количества средств, поэтому на данном этапе нет возможности нанять и организовать техническую поддержку пользователей. В дальнейшем планируется это сделать.

Система автоматизированной генерации расписания

Компоновка продукта

Версия 1.0

**История Ревизий**

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версия** | **Описание** | **Автор** |
| 31.05.2020 | 1.0 | Первоначальная версия | Павлов Данила, Допира Валерия |

**1. Введение**

**1.1 Цель**

Сборка исполняемого модуля из объектных модулей.

**2 Модульная организация программы**

Это разделение программы на более-менее независимые части (модули), их независимое проектирование и трансляция.

**2.1 Иерархия**

Любая сложная система не обходится без иерархии, без нее большая система превращается в нечто аморфное, необозримое и слабо управляемое.

Логическая иерархия отражает логических единиц программы, таких как функции, классы, библиотеки. Физическая иерархия касается физических единиц, на которые разбивается текст программы: файл (модуль), проект. Естественно, что между ними существует взаимосвязь, но не жесткая, синтаксическая, а технологическая, соблюдаемая программистом. Иерархия программных единиц имеет три уровня:

* элементом самого нижнего уровня является функция (в объектно-ориентированном программировании – метод класса). Это автономная синтаксическая единица языка. В традиционной технологии структурного программирования под модульным программированием понимают именно это: представление программы в виде системы взаимодействующих функций;
* несколько функций, объединенных общем описанием обрабатываемых ими структур данных, составляют библиотеку функций (эквивалент в ООП - класс). Все это – элементы логической иерархии. В физическом представлении им соответствует модуль (в интегрированных, закрытых системах) или файл исходного текста. Особенность модульного программирования в том и состоит, что отдельные модули могут разрабатываться, транслироваться и частично отлаживаться отдельно друг от друга. Но для этого им могут потребоваться описания интерфейсов взаимодействия (в Си – заголовочные файлы);
* вся программа в целом образуют проект. В интегрированных системах проект и все его модули могут быть представлены одним файлом. В традиционных системах программирования (к ним относится и С/С++) проект состоит из файлов исходного текста – модулей, файла проекта, содержащего список модулей, настройки транслятора и т.п., а также вспомогательных файлов. В этом случае под проект отводится отдельная папка.

**2.2 Фазы трансляции и выполнения программы**

Подготовка программы начинается с редактирования файла, содержащего текст этой программы, который имеет стандартное расширение для данного языка. Затем выполняется его трансляция (компиляция), которая включает в себя несколько фаз: препроцессор, лексический, синтаксический, семантический анализ, генерация кода и его оптимизация. В результате трансляции получается объектный модуль. Файл объектного модуля имеет стандартное расширение obj. Компоновка (сборка) программы заключается в объединении одного или нескольких объектных модулей программы и объектных модулей, взятых из библиотечных файлов, содержащих стандартные функции. В результате, будет получена исполняемая программа в виде файла, называемый загрузочный модуль или программный файл. В Windows стандартное расширение - .exe, а в Linux исполняемый файл определяется не по расширению, а по специальному флагу исполняемости. Полученный программный файл затем загружается в память и выполняется.

При модульном проектировании весьма важна разница между определением и объявлением объектов программы (переменных, функций, методов, классов). Определение переменной или функции – это фрагмент программы, в котором полностью задано содержание объекта и по которому происходит его трансляция во внутреннее представление. Объявление только упоминает объект языка и перечисляет его свойства, если он недоступен в данной точке программы. С учетом раздельного размещения определений и объявлений в проекте модульной Си-программы присутствуют три вида файлов (модулей):

* файлы исходного текста (с расширением - cpp), содержащие определения переменных, функций, методов;
* заголовочные файлы (с расширением - h), содержащие объявления для соответствующих файлов исходного текста;
* объектные модули (с расширением – obj), полученные в результате независимой трансляции файлов исходного текста.

Назначение заголовочных файлов заключается в том, что содержащиеся в них объявления позволяют сформировать правильный программный код для обращения к объекту языка, который определен в другом модуле. Для обращения к такому внешнему объекту необходимо подключить соответствующий заголовочный файл с его объявлением директивой include. То же самое касается данных и функций, содержащихся в библиотеках и библиотечных классах.

Для С++ это заголовочные файлы .h и файлы исходных коов .cpp. Также из-за того, что программа создана в виде проекта с использованием фреймворка Qt Creator, то есть еще файлы проекта с расширением .pro. Разработанный программный продукт должен поддерживаться на операционных системах Linux и Windows.

**2.3 Модульное программирование, компоновка**

При независимой трансляции модулей получается объектный модуль, содержащий часть программы во внутреннем представлении, а также информацию о некоторых элементах программы в исходном (символьном) виде:

* программный код, использующий в своей работе только объекты языка (типы данных, переменные, функции), определенные в текущем модуле, полностью переводится во внутреннее (двоичное) представление;
* если объект языка допускает внешний доступ из других модулей, то в объектом модуле создается точка входа, содержащая его имя и внутренний адрес в пространстве объектного модуля;
* при трансляции обращения к внешнему объекту языка объявление, полученное из заголовочного файла позволяет сформировать программный код для обращения к нему. Но все равно неизвестным остается его адрес. Поэтому вместо адреса транслятор оставляет внешнюю ссылку, содержащую исходное (символическое) имя объекта.

Библиотека объектных модулей - это файл (библиотечный файл), содержащий набор объектных модулей и собственный внутренний каталог. Объектные модули библиотеки извлекаются из нее целиком при наличии в них требуемых внешних функций и переменных и используются в процессе компоновки программы.

Компоновка (редактирование связей) - это процесс сборки программы из объектных модулей и библиотек, который включает в себя:

* объединение адресных пространств отдельных модулей в единое адресное пространство программного файла
* соединение внешних ссылок и соответствующих им точек входа
* при отсутствии необходимых точек входа для внешних ссылок их поиск производится в указанных библиотечных файлах. Если точка входа найдена в библиотеке объектных модулей, то весь объектный модуль, содержащий эту точку, компонуется в программу и для него повторяется описанный выше процесс.