ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ИРКУТСКОЙ ОБЛАСТИ «ЧЕРЕМХОВСКИЙ ГОРНОТЕХНИЧЕСКИЙ КОЛЛЕДЖ ИМ. М.И. ЩАДОВА»

РАССМОТРЕНО
на заседании ЦК
«Информатики и ВТ»
«31» июнь 2022 г.
Протокол № 10
Председатель: Окладникова Т.В.

УТВЕРЖДАЮ
И.о. зам. директора по УР
О.В. Папанова
«15» июнь 2022 г.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для выполнения
практических (лабораторных) работ студентов
по профессиональному модулю

ПМ.03 Ревьюирование программных продуктов
программы подготовки специалистов среднего звена

09.02.07 Информационные системы и программирование

Разработал преподаватель:
Коровина Н.С.

2022 г.

СОДЕРЖАНИЕ

1.ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Методические указания по выполнению практических (лабораторных) работ по профессиональному модулю **ПМ.03 Ревьюирование программных продуктов** предназначены для студентов специальности **09.02.07 «Информационные системы и программирование»,** составлены в соответствии с рабочей программой профессионального модуля **ПМ.03 Ревьюирование программных продуктов** и направлены на достижение следующих целей:

– овладение обучающимися видом профессиональной деятельности

1. Осуществлять ревьюирование программного кода в соответствии с технической документацией.
2. Выполнять измерение характеристик компонент программного продукта для определения соответствия заданным критериям.
3. Производить исследование созданного программного кода с использованием специализированных программных средств с целью выявления ошибок и отклонения от алгоритма.
4. Проводить сравнительный анализ программных продуктов и средств разработки, с целью выявления наилучшего решения согласно критериям, определенным техническим заданием.

– формирование у обучающегося умений осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития;

Методические указания являются частью учебно-методического комплекса по профессиональному модулю **ПМ.03 Ревьюирование программных продуктов** и содержат задания, указания для выполнения практических работ, теоретический минимум и т.п. Перед выполнением практической работы каждый студент обязан показать свою готовность к выполнению работы:

- ответить на теоретические вопросы преподавателя.

По окончанию работы студент оформляет отчет в тетради и защищает свою работу.

В результате выполнения полного объема практических работ студент должен у**меть:**

- Работать с проектной документацией, разработанной с использованием графических языков спецификаций.
- Применять стандартные метрики по прогнозированию затрат, сроков и качества. Определять метрики программного кода специализированными.
- Выполнять оптимизацию программного кода с использованием специализированных программных средств. Использовать методы и технологии тестирования и ревьюирование кода и проектной документации.
- Проводить сравнительный анализ программных продуктов. Проводить сравнительный анализ средств разработки программных продуктов. Разграничивать подходы к менеджменту программных проектов.

При проведении практических работ применяются следующие технологии и методы обучения:

1. проблемно-поисковых технологий
2. тестовые технологии

Правила выполнения практических работ:

1. Запомните порядок проведения практических (лабораторной) работ, правила их оформления.
2. Изучите теоретические аспекты практической (лабораторной) работы
3. Выполните задания практической (лабораторной) работы.
4. Оформите отчет в тетради.

Требования к рабочему месту:

- посадочные места по количеству студентов,
- рабочее место преподавателя,
- дидактическое обеспечение дисциплины:
- сборник заданий для практической (лабораторной) работы студентов

Технические средства обучения:

- Интерактивная доска, компьютер, диапроектор.
- Аппаратное обеспечение компьютеров:

Материнская плата GIGABYTE B450M DS3H

Системная плата совместима с процессорами от AMD. Она поддерживает сокет AM4, этот параметр необходимо учитывать при выборе подходящего чипа. Для доступа в Интернет применяется адаптер RealtekGbE с максимальной скоростью соединения 1000 Мбит/с. Обработкой звука занимается адаптер Realtek ALC887, он поддерживает схему 7.1 для объемного и качественного звучания.

2) Процессор AMD Ryzen 5 1600

Процессоры серии Ryzen – одни из наиболее мощных в линейке от AMD.

Модель имеет архитектуру Zen, ядро Summit Ridge и техпроцесс в 14 нм. Работает устройство с использованием 6 ядер. Диапазон частот 3200–3600 МГц сочетается со множителем 32. Двухканальная память модели принадлежит типу DDR4.

3) Видеокарта AMD Radeon Pro WX 2100

Видеокарта AMD RadeonPro WX 2100 относится к профессиональному классу. Частота работы видеочипа равна 1219 МГц. Установлена скоростная память GDDR5 с эффективной частотой 6000 МГц и пропускной способностью 96 ГБ/с. Максимальное энергопотребление адаптера – лишь 50 Вт.

4) 2 ТБ Жесткий диск Seagate 5900 SkyHawk

В качестве интерфейса подключения изготовители решили применить высокопродуктивный SATA III, благодаря чему скорость обмена данными с другими компонентами ПК может достигать 6 Гбит/с – огромная пропускная способность.

Передача данных осуществляется на скорости, максимум которая может

равняться 180 Мбайт/с.

Оперативнаяпамять AMD Radeon R7 Performance Series 8 ГБ

В 8-гигабайтный комплект входят два 4-гигабайтных модуля. Тип памяти – DDR4. Использует тактовую частоту 2666 МГц. Пропускная способность памяти равна 21300 МБ/с. Помимо тактовой, устройство может использовать другие частоты. Минимально допустимая частота – 1600 МГц. Модули характеризуются таймингами 16-18-18-35. Напряжение питания памяти, равное 1.2 В, соответствует стандартному показателю для DDR4.

- компьютер с лицензионным программным обеспечением и мультимедиа проектор (преподавательский);
- компьютеры с лицензионным программным обеспечением;

Критерии оценки:

Оценки «5» (отлично) заслуживает студент, обнаруживший при выполнении заданий всестороннее, систематическое и глубокое знание учебно - программного материала, учения свободно выполнять профессиональные задачи с всесторонним творческим подходом, обнаруживший познания с использованием основной и дополнительной литературы, рекомендованной программой, усвоивший взаимосвязь изучаемых и изученных дисциплин в их значении для приобретаемой специальности, проявивший творческие способности в понимании, изложении и использовании учебно- программного материала, проявивший высокий профессионализм, индивидуальность в решении поставленной перед собой задачи, проявивший неординарность при выполнении практического задания.

Оценки «4» (хорошо) заслуживает студент, обнаруживший при выполнении заданий полное знание учебно- программного материала, успешно выполняющий профессиональную задачу или проблемную ситуацию, усвоивший основную литературу, рекомендованную в программе, показавший систематический характер знаний, умений и навыков при выполнении теоретических и практических заданий по профессиональному модулю ПМ.06 Сопровождение информационных систем.

Оценки «3» (удовлетворительно) заслуживает студент, обнаруживший при выполнении практических и теоретических заданий знания основного учебно-программного материала в объеме, необходимом для дальнейшей учебной и профессиональной деятельности, справляющийся с выполнением заданий, предусмотренных программой, допустивший погрешности в ответе при защите и выполнении теоретических и практических заданий, но обладающий необходимыми знаниями для их устранения под руководством преподавателя, проявивший какую-то долю творчества и индивидуальность в решении поставленных задач.

Оценки «2» (неудовлетворительно) заслуживает студент, обнаруживший при выполнении практических и теоретических заданий проблемы в знаниях основного учебного материала, допустивший основные принципиальные ошибки в выполнении задания или ситуативной задачи, которую он желал бы решить или предложить варианты решения, который не проявил творческого подхода, индивидуальности.

В соответствии с учебным планом программы подготовки специалистов среднего звена по специальности **09.02.07 «Информационные системы и программирование»** и рабочей программой на практические работы по профессиональному модулю **ПМ.03 Ревьюирование программных продуктов** отводится 34 часов.

2. ПЕРЕЧЕНЬ ПРАКТИЧЕСКИХ РАБОТ

№ п/п	Название практической работы (указать раздел программы, если это необходимо)	Количество часов
МДК 03.01 Моделирование и анализ программного обеспечения		
1	Практическая работа №1. «Организация работы в команде». Причины создания команды. Принципы объединения ролей. Процесс управления рисками (практический пример).	2
2	Практическая работа №2. «SharePoint Services. Создание учетных записей портала. Назначение прав доступа к библиотекам. Добавление содержимого. Разработка технической документации. Размещение документации и её обсуждение на портале SharePoint. Организация дискуссии технического задания. Создание кризиса – митинга (дискуссия, посвященная решению кризисной проблемы)	2
3	Практическая работа №3. «Создание и изучение возможностей репозитория проекта». «Экспорт настроек в командной среде разработки».	2
4	Практическая работа №4. «Разработка требований к программному приложению»	2
5	Практическая работа №5. «Планирование итераций». «Моделирование интерфейса пользователя». «Работа с базой данных в автономном режиме».	2
6	Практическая работа №6. «Управлении этапом разработки кода программных компонентов». «Построение приложений в командном проекте»	2
7	Практическая работа №7. «Создания тестовых случаев в командном проекте». «Формирование отчетов».	2
8	Практическая работа №8. «Сравнительный анализ офисных пакетов». «Сравнительный анализ браузеров». «Сравнительный анализ средств просмотра видео».	2
МДК 03.02. Управление проектами		
9	Практическая работа №1. «Профилирование приложений в Visual Studio». «Использование метрик программного продукта»	2
10	Практическая работа №2. «Офускация кода». «Проверка целостности программного кода»	2
11	Практическая работа №3. «Анализ потоков данных». « Использование метрик стилистики»	2
12	Практическая работа №4. «Выполнение измерений характеристик кода в среде Visual Studio».	2
13	Практическая работа №5. «Выполнение измерений характеристик кода в среде (например, Eclipse C/C++ и др.)»	2
14	Практическая работа №6. «Определение проекта и его границ». «Организационные структуры». «Методологии ведения проектов».	2
15	Практическая работа №7. «Построение команды проекта» . «Начало и завершение проекта»	2
16	Практическая работа №8. «Коммуникации в проекте».	2

	«Построение иерархической структуры работ проекта» .	
17	Практическая работа №9. «Смета затрат на разработку и реализацию проекта»	2
Итого		**34**

3. СОДЕРЖАНИЕ ПРАКТИЧЕСКИХ РАБОТ

МДК 03.01 Моделирование и анализ программного обеспечения

Практическая работа №1.

«Организация работы в команде».

Причины создания команды. Принципы объединения ролей.

Процесс управления рисками.

Цель: рассмотреть организацию в команде, причины создания команды, принципы объединения ролей.

Задание 1. Изучить пример построения матрицы ответственности и построить матрицу ответственности для своего проекта.

Построение матрицы ответственности

1. Перечислить основные работы проекта. По вертикали в матрице отражаются только основные работы проекта (не ниже уровня 2-3 ИСР), но с достаточной степенью детализации для обеспечения возможности указывать разные роли, необходимые для выполнения этих работ. Когда речь идет о крупных проектах и программах, может возникнуть необходимость разработать несколько матриц ответственности с различной степенью детализации.
2. Перечислить группы/роли внутри проектной команды. По горизонтали в матрице перечисляются группы/ роли внутри проектной команды. Обратите внимание на то, что в матрице ответственности группы/роли, а не имена и фамилии отдельных членов коллектива. Персональное закрепление проектных работ производится позднее, на этапе разработки расписания проекта.
3. Закодировать матрицу ответственности. С помощью кодов в ячейках на пересечении соответствующих столбцов с ролями и строк с работами проекта указать степень участия, формальные полномочия и распределение ответственности за выполнение каждой операции. Четкое указание разных уровней формальных полномочий бывает особенно полезно в ситуации, когда множество членов проектной команды желает предъявить особые требования к проекту.

Условные обозначения матрицы ответственности (RACI)		
Обозначение	Расшифровка	Описание
Исп. (R)	Исполнитель (Responsible)	Несет ответственность за непосредственное исполнение задачи. К каждой задаче должно быть приписано не менее одного исполнителя
Утв. (A)	Утверждающий (Accountable)	Отвечает за конечный результат перед вышестоящим руководством. На каждую работу должен быть назначен строго один подотчетный
Согл. (C)	Согласующий (Consulted)	Согласует принимаемые решения, взаимодействие с ним носит двусторонний характер
Н. (I)	Наблюдатель	Его информируют об уже принятом решении,

	(Informed)	взаимодействие с ним носит односторонний характер

. Распределение функциональных обязанностей команды управления проектом

Функциональные обязанности	Куратор проекта (Спонсор)	Руководитель проекта	Архитектор системы	Администратор проекта
Планирование				
Разработка и периодическая актуализация плана		+	+	
Утверждение плана	+			
Управление командой проекта				
Назначение сотрудника на роль Руководителя проекта	+			
Формирование команды проекта		+		
Определение квалификационных требований и состава рабочих групп специалистов по функциональности ИС			+	
Обеспечение выделения необходимых ресурсов для выполнения проекта	+			
Непосредственное руководство Командой проекта		+		
Формирование предложений по стимулированию Команды проекта	+			
Обеспечение стимулирования Команды проекта	+			
Организация выполнения работ				
Организация взаимодействия с Заказчиком и обеспечение всех необходимых коммуникационных связей с другими участниками проекта		+		
Организация подготовки, согласования и утверждения всей технической документации, необходимой для создания ИС в рамках проекта			+	
Организация, проведение и документирование процедур передачи Заказчику		+	+	

разработанной ИС				
Рассмотрение и утверждение регламентирующих документов, необходимых для организации и выполнения проекта	+			
Ведение организационно-распорядительной и отчетной документации. Поддержание в актуальном состоянии списка команды проекта				+
Обеспечение команды проекта необходимыми информационными материалами				+
Материально-техническое и хозяйственное обеспечение команды проекта				+
Контроль хода выполнения проекта				
Организация и проведение совещаний по обсуждению хода работ проекта		+		
Подготовка и предоставление Куратору отчетов о ходе работ проекта		+		
Получение и анализ сводной отчетности о ходе реализации проекта	+			
Контроль соответствия результатов проекта Техническому заданию на разработку ИС			+	
Согласование фактических трудозатрат специалистов при исполнении проекта		+	+	

На коды, используемые в матрице ответственности, каких-либо ограничений не существует, но наибольшее распространение получил метод RACI (Responsible (R), Accountable (A), Consulted (C), Informed(I)), в котором приведено описание соответствующих кодов.

4. Инициировать использование матрицы и включить процедуру использования матрицы ответственности в документ "План управления проектом".

После утверждения матрицы ответственности все дальнейшие изменения в ней должны проходить через процедуру интегрированного управления изменениями при участии авторов первоначальной версии.

Преимущество использования структурированного подхода к изменению матрицы ответственности состоит в том, что руководитель проекта получает актуальный документ, на который он может ссылаться при возникновении тех или иных спорных ситуаций, касающихся распределения полномочий в проекте.

Задание 2. Создать иерархическую структуру работ и закрепить функции и полномочия в проекте на основе примера.

ИСР – это согласованная с результатами поставки иерархическая декомпозиция работ, которые команда проекта должно выполнить для достижения целей проекта и создания оговоренных результатов поставки.

С ее помощью структурируется и определяется все содержание проекта.

ИСР подразделяет работы проекта на более мелкие и более управляемые части, где на каждом более низком уровне ИСР дается более детальное определение проектных работ.

Пример элемента ИСР

№ (отражает уровень мероприятия)	Мероприятие (краткое описание)	Цель / Планируемый результат	Ответственный и исполнители	Сроки выполнения
1.	**Планирование проекта**			1.08.08.-15.08.08
1.1.	Прием заказа	Заявка по форме	Секретарь	1.08.08.
1.2.	Назначение руководителя проекта и команды проекта	Приказ	Директор	2.08.08
1.3.	Анализ заявки и предварительная проработка проекта	Отчёт по результатам анализа	руководитель проекта и команда проекта	2.08.08 – 5.08.08
1.4.	Разработка плана проекта	План проекта	Руководитель проекта	5.08.08 – 15.08.08
1.4.1.	Проведение маркетингового исследования	Отчёт по исследованию	Маркетолог, руководитель проекта и команда проекта	5.08.08 – 10.08.08
1.4.2	Разработка плана проекта	План проекта	Руководитель проекта	10.08.08 – 14.08.08
1.4.3	Утверждение плана проекта у заказчика	Утверждённый план проекта	Директор	15.08.08

Итог работы: отчет, защита

Практическая работа № 2

«**SharePoint Services.** Создание учетных записей портала. Назначение прав доступа к библиотекам. Добавление содержимого. Разработка технической документации. Размещение документации и её обсуждение на портале SharePoint. Организация дискуссии технического задания. Создание кризиса – митинга (дискуссия, посвященная решению кризисной проблемы)

Цель Изучить:

- способы создания учетных записей портала.
- назначение прав доступа к библиотекам.
- добавление содержимого.
- разработку технической документации.
- размещение документации и её обсуждение на портале SharePoint.

- организацию дискуссии технического задания.
- создание кризиса – митинга (дискуссия, посвященная решению кризисной проблемы)

Задание:

1. Создать портал SharePoint.
2. Добавить новых пользователей портала SharePoint
3. На портале создать: библиотеки документов и рисунков, список, доску обсуждений.
4. Определить права доступа к созданным библиотекам и списку.
5. Добавить материалы в библиотеки.
6. Создать дочерний сайт.
7. Настроить параметры времени портала, изменить тему. Прочие настройки – по усмотрению.

Итог работы: отчет

Практическая работа №3.

«Создание и изучение возможностей репозитория проекта».
«Экспорт настроек в командной среде разработки».

Цель:

 получить опыт практической работы с системой контроля версий на примере Tortoise SVN; получить навыки создания репозитория.

Задание 1

1. Создание хранилища (репозитория) SVN

1.1. На одном из локальных дисков создайте папку, в которой в дальнейшем будет находится репозиторий SVN для проекта. Лучше, чтобы эта папка находилась на достаточно надежном носителе. Пусть это будет папка E:\MyRepository\MyProject.

1.2. Нажмите правой кнопкой мыши на созданной папке и выберите команду TortoiseSVN\Create repository here.

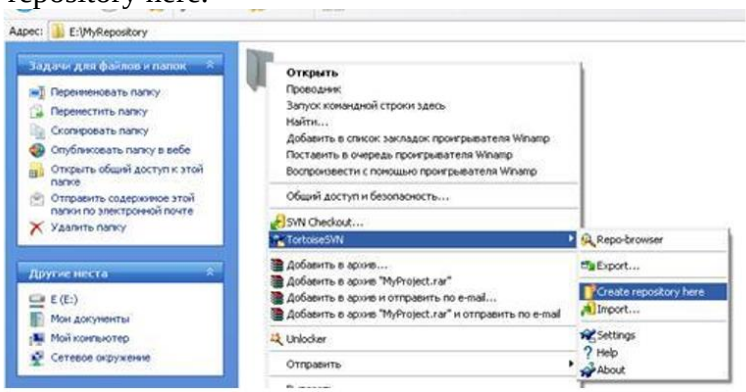


Рис.1.2 Создание репозитория

1.3. Появится окно с сообщением, что репозиторий успешно создан. После чего нажмите кнопку ОК.

1.4.

Рис.1.3 Окно сообщения об успешном создании хранилище

1.5. Далее появятся файлы как показано ниже на рисунке/

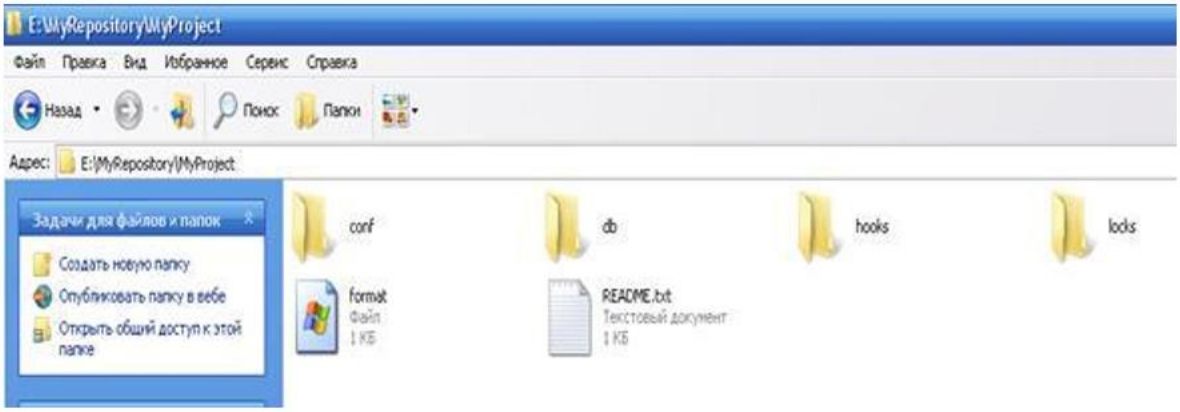


Рис.1.4 Структура репозитория

Примечание: Хранилище будет создано внутри новой папки. Не редактируйте эти файлы самостоятельно!!! В случае возникновения ошибок убедитесь, что папка пуста и доступна для записи.

2. Импорт данных в хранилище Предполагая, что хранилище уже у вас есть, и вы желаете добавить в него новую папку со всей еѐ структурой, просто выполните следующие шаги:
2.1. При помощи обозревателя хранилища (TortoiseSVN → Repo-browser) создайте новую папку проекта под именем 123 непосредственно в хранилище.

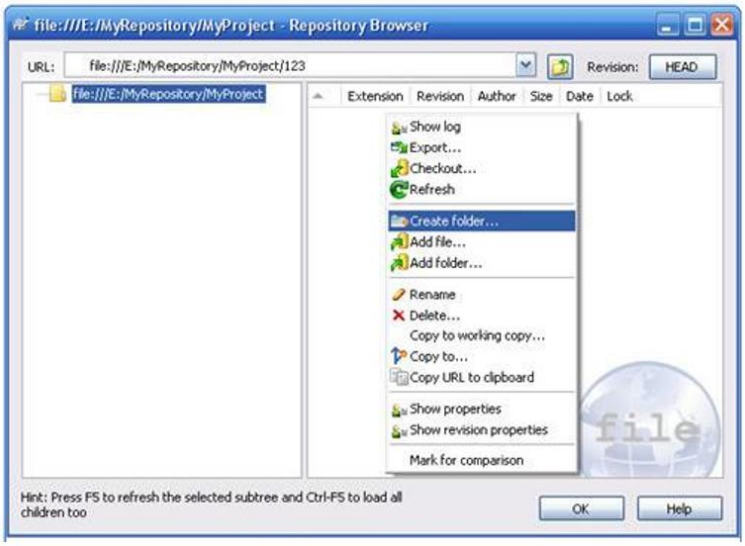


Рис.1.5 Обозреватель хранилища

2.2. Откройте общий доступ хранилище вкладе Доступ пометив флажком в Открыть общий доступ к этой папке и Разрешить изменение файлов в сети.

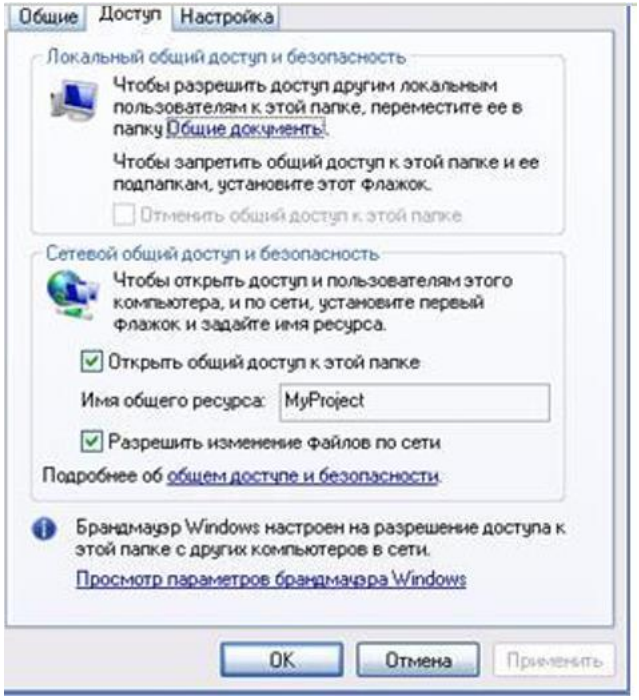


Рис.1.6 Открытие общего доступа к репозиторий.

3. Извлечение рабочей копии

3.1. Для получения рабочей копии вам надо произвести извлечение из хранилища. Выберите в Проводнике Windows папку, в которой хотите поместить вашу рабочую копию. Сделайте правый щелчок для вызова контекстного меню и выберите команду TortoiseSVN -> Извлечь ...,(TortoiseSVN → Checkout...,)

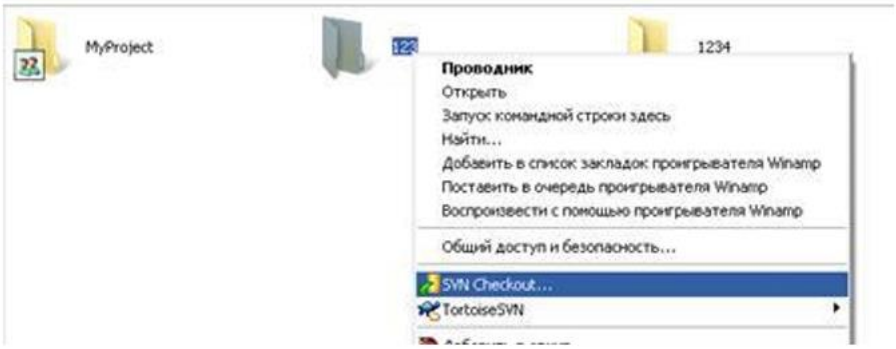


Рис.1.7 Меню для извлечения рабочей копии

3.2. Появившемся диалоговом окне Checkout в поле URL of Repository вставьте путь к репозиторий и нажимаем ОК, далее еще раз нажимаете ОК. Для того что вставить выделите репозиторий – папку и с помощью правой кнопки мыши откройте обозревателя хранилища (TortoiseSVN → Repo-browser) в поле URL скопируйте путь. В поле Checkout directory автоматически появляется путь к рабочей копии, т. е. к папке.

Рис.1.8 Путь к репозиторий

3.3. Выделите рабочую копию, т. е. папку под именем 123 и нажмите F5, либо правой кнопкой мыши нажмите обновить.

3.4. Повторите действия 3.1–3.3 для рабочей копии 1234.

Задание 2:

1. Аналогично проделать те же действие, но проделав эти же действие на двух разных компьютерах.
2. В одном компьютере например, комп1 создаете репозиторию и одну рабочую копию, т. е. папку под именем 123.
3. Во втором компьютере например, комп2 создаете только рабочую копию, т. е. папку под именем 1234.
4. Во втором компьютере соедините рабочую копию с репозиторием указывая путь file:///ip-компьютера/название_репозитория.
5. В комп1 создайте любой файл и отправьте его на комп2. Таким образом проделав 5 ревизий.
6. Чтобы узнать IP- компьютера нажмите пуск →выполнить→ cmd, либо C:\Windows\System32\cmd.exe /k %windir%\system32\ipconfig.exe
7. Откроется окно командой строки. Введите ipconfig, где вы увидите примерное IP-компьютера: 192.168.8.xxx

Итог работы: файлы, отчет

Практическая работа №4.

«Разработка требований к программному приложению»

Цель: ознакомиться с правилами написания технического задания

Задание 1. Изучить теоретические сведения

Теоретические сведения

1. Техническое задание оформляют в соответствии с ГОСТ 19.106—78 на листах формата А4 и А3 по ГОСТ 2.301—68, как правило, без заполнения полей листа. Номера листов (страниц) проставляют в верхней части листа над текстом.
2. Лист утверждения и титульный лист оформляют в соответствии с ГОСТ 19.104—78. Информационную часть (аннотацию и содержание), лист регистрации изменений допускается в документ не включать.
3. Для внесения изменений и дополнений в техническое задние на последующих стадиях разработки программы или программного изделия выпускают дополнение к нему. Согласование и утверждение дополнения к техническому заданию проводят в том же порядке, который установлен для технического задания.

Задание 2. Составьте техническое задание, на следующие условия:
Описание предметной области:
Организация: «House»
Функциональные особенности организации:
Данная организация занимается гостиничным бизнесом.
Задачей сотрудников организации является отслеживание финансовой стороны работы организации: «House».
Деятельность организована следующим образом: гостиница представляет номера клиентам на определенный срок.

Техническое задание должно содержать следующие разделы:

- введение;
- наименование и область применения;
- основание для разработки;
- назначение разработки;
- технические требования к программе или программному изделию;
- технико-экономические показатели;
- стадии и этапы разработки;
- порядок контроля и приемки;
- приложения.

Итог работы: техническое задание

Практическая работа №5.
«Планирование итераций». «Моделирование интерфейса пользователя».
«Работа с базой данных в автономном режиме».

Цель: освоение средства разработки программного кода MS Visual Studio для программирования алгоритмов внутренней сортировки, изучение и освоение применения процедуры рефакторинга для улучшения программного кода.

Задание:
1. Создать консольное приложение в MS Visual Studio C# и выполнить сборку программы решения типовой задачи обработки данных о сотрудниках предприятия с использованием односвязных списков. Откомпилировать и построить приложение. При обнаружении компилятором синтаксических ошибок идентифицировать их и устранить.
2. На базе созданного приложения реализовать заданную функциональность, сформулированную в техническом задании при выполнении лабораторной работы №3.

Добавить в класс List методы SortData1(), SortData2(),SortData3(), который реализует задачу сортировки записей заданными методами приложение. При обнаружении компилятором синтаксических ошибок идентифицировать их и устранить.

3. Произвести оценку сложности разработанных методов сортировки. В качестве критерия Сложности использовать суммарное операторов метода.

Итог работы: файл, защита

Практическая работа №6.

«Управлении этапом разработки кода программных компонентов». «Построение приложений в командном проекте»

Цель: рассмотреть Управлении этапом разработки кода программных компонентов». «Построение приложений в командном проекте

Задание:

В соответствии с подготовленным техническим заданием выполнить разработку спецификаций на программный продукт, которые должны включать:

- спецификации процессов;
- словарь терминов;
- диаграммы переходов состояний;
- диаграммы потоков с детализацией.

2.Оформить отчет.

Содержание отчета:

-тема практической работы;
- цель работы;
- задание на практическую работу;
- разработанные спецификации процессов;
- словарь терминов;
- диаграммы переходов состояний;
- диаграммы потоков с детализацией;
- выводы по проделанной работе.

Итог работы: отчет, файлы

Практическая работа №7.

«Создания тестовых случаев в командном проекте».
«Формирование отчетов».

Цель: рассмотреть тестовые случае в командном проекте, формирование отчетов.

Задание:

1. Проведите сравнительный анализ состава рабочих элементов в шаблонах MSFforCMMIProcessImprovement 6.0 и MSFforAgileSoftwareDevelopment 6.0.
2. Проанализируйте процесс управления рисками проектов в методологии MSFforCMMIProcessImprovement 6.0.
3. Проанализируйте процесс управления командами проектов в методологии MSFforCMMIProcessImprovement 6.0.
4. Написать отчет в тетради

Итог работы: отчет

Практическая работа №8.

«Сравнительный анализ офисных пакетов».

«Сравнительный анализ браузеров». «Сравнительный анализ средств просмотра видео».

Цель:

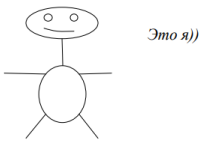
- получение офисных пакетов Microsoft и OpenOffice;
- изучение принципов и получение практических навыков;
- изучение содержания офисных пакетов;
- получение практических навыков выполнения основных операций в среде офисных пакетов MS и OO;
- получение навыков выполнения анализа

Задание:

1. Создать документ отображающий содержание практической работыв среде MS Word.
2. Создать таблицу с данными о составе практических работ, их продолжительности.
3. Создать рисунок аналогичный приведѐнному ниже.
4. Сохранить документ в формате «.doc».
5. Открыть документ в среде OpenOffice.
6. Сделать вывод о полученном результате.

Вывод должен передавать сохранность вида текста, таблицы, рисунка.

7. Выполнить аналогичную работу в среде OpenOffice.
8. Сделать вывод о полученном результате.



Задание 2:

1. Загрузите страницу https://docs.microsoft.com/ruru/aspnet/mvc/pluralsight с помощью браузера Google Chrome.
2. Выполните функции:

a. чтения теста в слух;
b. перевода;
c. просмотра видео.

3. Выполните данные функции, если возможно, с помощью браузеров Mozilla, MS Edge.
4. Отобразите на диаграмме вариантов использования функции, выполняемые браузерами.
5. Сделайте вывод по особенностям выполнения функций.

Итог работы: отчет, файлы

МДК 03.02. Управление проектами

Практическая работа №1.

«Профилирование приложений в Visual Studio».
«Использование метрик программного продукта»

Цель: Научиться использовать Microsoft Visual Studio для разработки программ на языке C++. Получить практические навыки работы со средой визуальной разработки программ.

Задание 1. Создание приложения

Запускаем Microsoft Visual C++ .

После запуска системы мы увидим начальный пользовательский интерфейс, показанный на рис. 1.1.

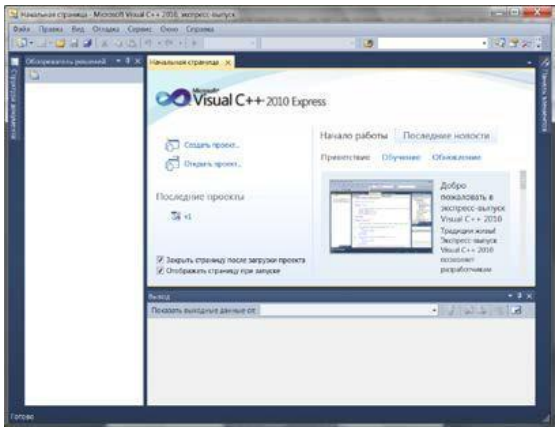


Рис. 1.1. Фрагмент стартовой страницы системы Visual Studio

Для создания приложения, необходимо в пункте меню File выполнить команду New Project (Новый проект). В появившемся окне New Project в левой колонке находится список установленных шаблонов (Installed Templates). Среди них — шаблоны языков программирования, встроенных в Visual Studio, в том числе Visual Basic, Visual C#, Visual C++, Visual F# и др. Нам нужен язык Visual C++. В узле Visual C++ области типов проектов выберем среду CLR, а затем в области шаблонов (в средней колонке) выберем шаблон (Templates) Windows Forms Application Visual C++.
Теперь введем имя проекта (Name) v4 и щелкнем на кнопке OK, в результате увидим окно, представленное на рис. 1.2.

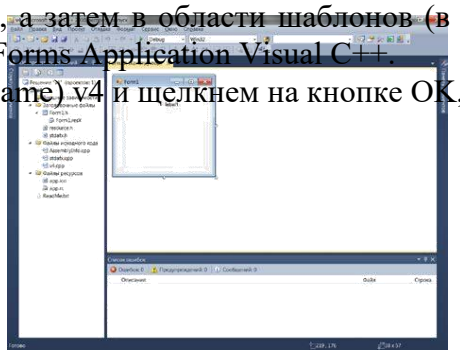


Рис. 1.2. Окно для проектирования пользовательского интерфейса

В этом окне изображена экранная форма — Form1. Первая программа будет отображать такую экранную форму, в которой будет что-либо написано, например «Microsoft Visual C++ 2010», также в форме будет расположена командная кнопка с надписью «Нажми меня». При нажатии кнопки будет появляться диалоговое окно с сообщением «Всем привет!» В программе четыре объекта: форму Form, надпись на форме Label, кнопка Button и диалоговое окно MessageBox с текстом «Всем привет!» (окно с приветом).
Добавить в форму названные элементы управления. Для этого понадобится панель элементов управления Toolbox (Панель управления), ее можно добавить, например, с помощью комбинации клавиш Ctrl+Alt+x или ViewToolbox. Итак, добавьте метку Label и кнопку Button в форму, дважды щелкая на этих элементах на панели Toolbox. А затем следует рас положить их примерно так, как показано на рис. 1.3.

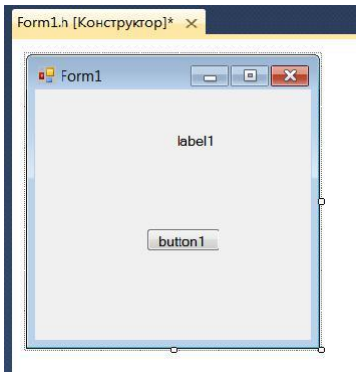


Рис. 1.3. Форма первого проекта

Каждый объект имеет свойства (properties- Свойства) . Свойств много, их можно увидеть, если щелкнуть правой кнопкой мыши в пределах формы и выбрать в контекстном меню команду Properties-Свойства, при этом появится панель свойств.
Для объекта label1 выбрать свойство Text и написать напротив этого поля «Microsoft Visual C++ 2010» (вместо текста label1). Для объекта button1 также в свойстве Text написать «Нажми меня».
Объекты не только имеют свойства, но и обрабатываются событиями. В задаче событием, которым управляем, является щелчок на командной кнопке. Для получения пустого обработчика этого события следует в свойствах кнопки button1 щелкнуть на значке молнии Events (события) и в списке всех возможных событий кнопки button1 выбрать двойным щелчком событие Click. После этого попадаем на вкладку программного кода Form1.h (см.
рис. 1.4).

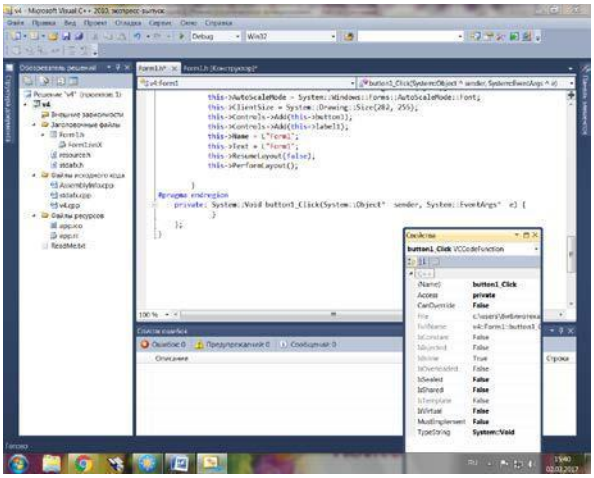


Рис 1.4 Вкладка программного кода

На вкладке Form1.h видно, что управляющая среда Visual C++ сгенерировала довольно таки много строк программного кода. В этом тексте уже можно найти те присваивания, которые сделали в панели свойств Properties. Например, для свойства Text кнопки Button управляющая среда назначила строку ≪Нажми меня≫: **this->button1->Text = L**"Нажми меня"**;**
Пустой обработчик события button1_Click:
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) { }

Здесь в фигурных скобках пишутся команды, подлежащие выполнению после щелчка на кнопке. В фигурных скобках обработчика события напишите:
MessageBox::Show("Всем привет!");
Теперь нажмите клавишу F5 и проверьте работоспособность программы (рис. 1.5).

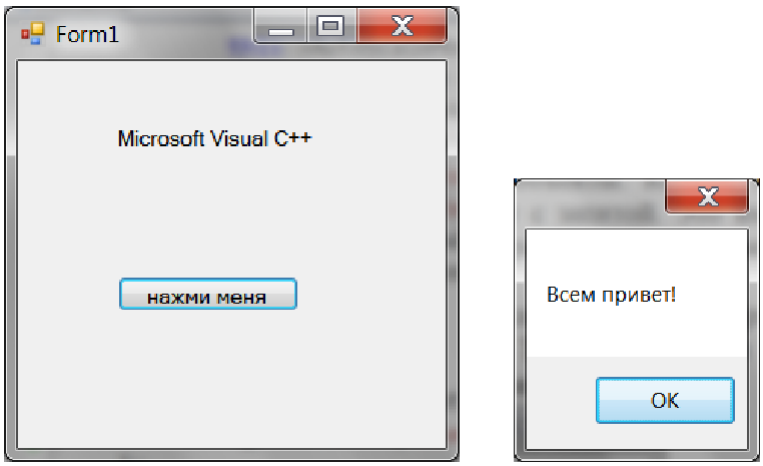


Рис. 1.5. Фрагмент работы программы

Задание 2. Обработка события MouseHover мыши
Событие MouseHover наступает тогда, когда пользователь указателем мыши «зависает» над каким-либо объектом, событие MouseHover происходит, когда указатель мыши наведен на элемент.

Таким образом, программа в данном примере должна содержать на экранной форме текстовую метку Label и кнопку Button. Метка должна отображать текст «Microsoft Visual C++ 2010»; при щелчке на командной кнопке, на которой попрежнему будет написано «Нажми меня», появится диалоговое окно с сообщением «Всем привет!» Кроме того, когда указатель мыши наведен на текстовую метку (то самое событие MouseHover), должно появиться диалоговое окно с текстом «Событие Hover».

Для решения этой задачи запустим Visual Studio 2010, щелкнем на пункте меню New Project. В появившемся окне New Project в левой колонке в узле Visual C++ выберем среду CLR, а затем в области шаблоны (в средней колонке) выберем шаблон (Templates) Windows Forms Application Visual C++. В качестве имени проекта введем имя Hover и щелкнем на кнопке OK.
В дизайнере формы из панели Toolbox перетащим на форму метку Label и кнопку Button, а затем немного уменьшим размеры формы на свое усмотрение. Теперь добавим три обработчика событий в программный код. Для этого в панели Properties следует щелкнуть на значке молнии (Events) и двойным щелчком последовательно выбрать событие загрузки формы Form_Load, событие «щелчок на кнопке button1_Click» и событие label1_MouseHover.
При этом осуществится переход на вкладку программного кода Form1.h, и среда Visual Studio 2010 сгенерирует три пустых

обработчика события (рис.1.6). Например, обработчик последнего события будет иметь вид:
private:System::Void lab el1_MouseHover(System::Object^ sender, System::EventArgs^ e){}

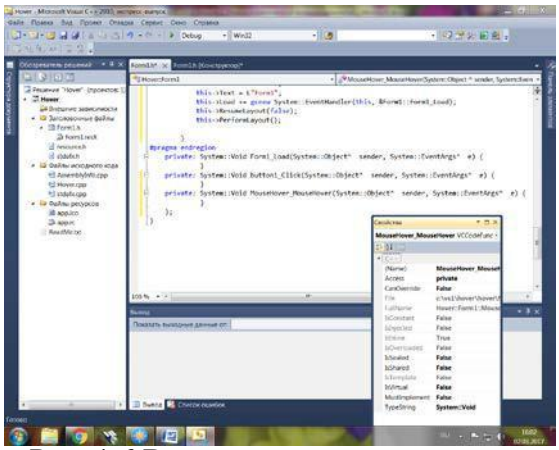


Рис. 1.6 Вкладка программного кода

Между фигурными скобками вставим вызов диалогового окна:
MessageBox::Show("Событие Hover!");
Теперь проверим возможности программы: нажимаем клавишу F5, «зависаем» указателем мыши над label1, щелкаем на кнопке button1. Все работает! (рис.1.7)

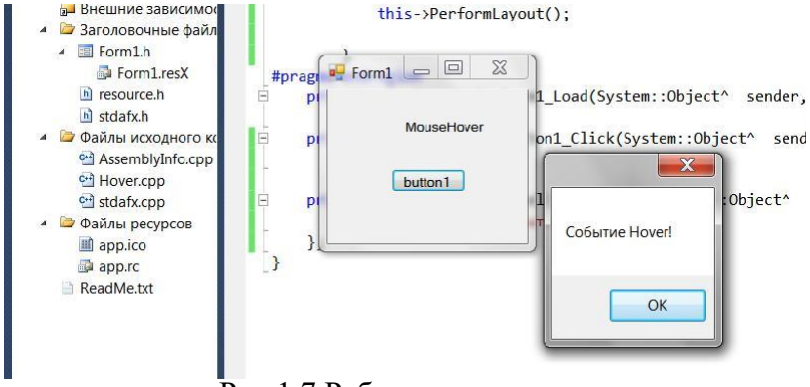


Рис.1.7 Работа приложения

Листинг. Фрагмент файла Form1.h, содержащего программный код с тремя обработчиками событий
// .....
// Программный код, расположенный выше, создан средой Visual Studio
// автоматически, поэтому автором не приводится
this->ResumeLayout(false);
this->PerformLayout();
}
#pragma endregion

```
//        Данная программа управляется тремя событиями. Событие загрузки формы
//      Form1_Load инициализирует надписи заголовка формы, текстовой метки
//       и кнопки. Событие щелчок на кнопке button1_Click вызывает появление
//       диалогового окна с текстом "Всем привет!". Событие, когда указатель
//      мыши наведен на метку, вызывает появление диалогового окна с текстом
//     "Событие Hover".
private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^
e)
{ // Обработка события загрузки формы: this->Text =
"Приветствие";
// или Form1::Text = "Приветствие"; label1->Text =
"Microsoft Visual C++ 2010"; button1->Text = "Нажми меня";
}
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{ // Обработка события щелчок на кнопке:
MessageBox::Show("Всем привет!");
}
private: System::Void label1_MouseHover(System::Object^ sender, System::EventArgs^ e)
{ // Обработка события, когда указатель мыши наведен на метку:
MessageBox::Show("Событие Hover!");
}
};
}
```

Контрольные вопросы

1. Из каких двух этапов состоит процесс проектирования программы Visual C++.
2. Что такое программа, основанная на диалоге.
3. Что такое Windows Forms Application.

Итог работы: файлы, защита

Практическая работа №2.
«Офускация кода». «Проверка целостности программного кода»

Цель: Научиться разрабатывать и реализовывать простейшие программы на языке VC++. Получить практические навыки работы по использованию интерфейса CLR. Научиться связывать переменные и методы с элементами диалогового окна.

Задание. Ввод данных через текстовое поле TextBox с проверкой типа методом TryParse
При работе с формой очень часто ввод данных организуют через элемент управления текстовое поле TextBox . Напишем типичную программу, которая вводит через текстовое поле число, при нажатии командной кнопки извлекает из него квадратный корень и выводит результат на метку Label. В случае ввода не числа сообщает пользователю об этом.

Решая сформулированную задачу, запускаем Visual Studio, выбираем пункт меню File-New -Project. В окне New Project в узле Visual C++ выберем среду CLR, а затем в области шаблоны выберем шаблон (Templates) Windows Forms Application Visual C++. В качестве имени проекта введем имя Корень и щелкнем на кнопке OK.

Далее из панели элементов управления Toolbox (если в данный момент вы не видите панель элементов управления, то ее можно добавить, например, с помощью комбинации клавиш Ctrl+Alt+x или меню View - oolbox) в форму с помощью указателя мыши перетаскиваем текстовое поле TextBox, метку Label и командную кнопку Button. Получить названные элементы на проектируемой экранной форме можно , также дважды щелкая указателем мыши на каждом элементе в панели Tools. Таким образом, в форме будут находиться три элемента управления. Расположим их на экранной форме.

Теперь следует изменить некоторые свойства элементов управления. Чтобы получить пустой обработчик загрузки формы, дважды щелкнем по проектируемой экранной форме. Сразу после этого мы попадаем на вкладку программного кода Form1.h. Здесь задаем свойствам формы (к форме обращаемся посредством ссылки this), кнопкам button1 и текстового поля textBox1, метке label1 следующие значения:

this->Text = "Извлечение квадратного корня"; button1->Text
= "Извлечь корень"; textBox1->Clear(); // Очистка текстового
поля label1->Text = nullptr; // или = String::Empty;

Нажмем клавишу F5 для выявления возможных опечаток, то есть синтаксических ошибок и предварительного просмотра дизайна будущей программы (рис.2.1).

Далее программируем событие button1 _Click — «щелчок мышью на кнопке Извлечь корень». Создать пустой обработчик этого события удобно, дважды щелкнув мышью на этой кнопке. Между двумя появившимися строчками программируем диагностику правильности вводимых данных, конвертирование строковой переменной в переменную типа Single и непосредственное извлечение корня (листинг).

Листинг. Фрагмент программы извлечения корня с проверкой типа методом TryParse

```
//     .............
//      Программный код, расположенный выше, создан средой Visual
Studio
//     автоматически, поэтому автором не приводится
this->ResumeLayout(false);
this->PerformLayout();
}
#pragma endregion
//      Программа вводит через текстовое поле число, при щелчке на командной
//        кнопке извлекает из него квадратный корень и выводит результат
//        на метку label1. В случае ввода не числа сообщает пользователю об
//      этом, выводя красным цветом предупреждение также на метку label1.
private: System::
Void Form1_Load(System::Object^ sender, System::EventArgs^ e)
{
button1->Text = "Извлечь корень"; label1-
>Text = String::Empty;
//     или label1->Text = nullptr;
this->Text = "Извлечение квадратного корня"; textBox1->Clear(); // Очистка
текстового поля textBox1->TabIndex = 0; // Установка фокуса в текстовом поле
}
private: System:: // Обработка щелчка на кнопке "Извлечь корень":
Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{
Single X; // - из этого числа будем извлекать корень
//     Преобразование из строковой переменной в Single: bool Число_ли =
Single::TryParse(textBox1->Text, System::Globalization::NumberStyles::Number,
System::Globalization::NumberFormatInfo::CurrentInfo, X);
//     Второй параметр - это разрешенный стиль числа (Integer,
//        шестнадцатеричное число, экспоненциальный вид числа и прочее).
//     Третий параметр форматирует значения на основе текущего языка
//     и региональных параметров из Панели управления - Язык и
//     региональные стандарты - число допустимого формата; метод

//     возвращает значение в переменную X if
(Число_ли == false)
```

```
{ // Если пользователь ввел не число: label1->Text =
"Следует вводить числа";
label1->ForeColor = Color::Red; // - цвет текста на метке return; // - выход из
процедуры
}
Single Y = (Single)Math::Sqrt(X); // - извлечение корня label1->ForeColor =
Color::Black; // - черный цвет текста на метке
label1->Text = String::Format("Корень из {0} равен {1:F5}", X, Y);
}
};
}
```

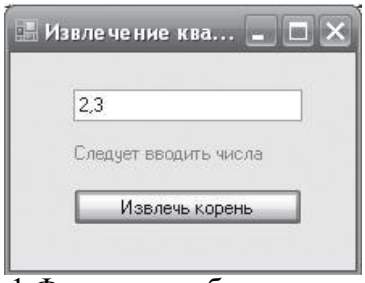


Рис.2.1 Фрагмент работы программы

Если пользователь ввел все-таки число, то будет выполняться следующий оператор извлечения квадратного корня Math::Sqrt(X). Математические функции Visual Studio являются методами класса Math. Их можно увидеть, набрав Math и введя так называемый *оператор разрешения области действия* (::). В раскрывающемся списке вы увидите множество математических функций: Abs, Sin, Cos, Min и т. д. и два свойства — две константы E = 2,718...
(основание натуральных логарифмов) и PI = 3,14... (число диаметров, уложенных вдоль окружности). Функция Math::Sqrt(X)возвращает значение типа double (двойной точности с плавающей запятой), которое мы *приводим* с помощью неявного преобразования (Single) к переменной одинарной точности.
Последней строчкой обработки события button1_Click является формирование строки label1->Text с использованием метода String::Format. Использованный формат ≪Корень из {0} равен {1:F5}≫ означает: взять нулевой выводимый элемент, то есть переменную X, и записать эту переменную вместо фигурных скобок; после чего взять первый выводимый элемент, то есть Y, и записать его вместо вторых фигурных скобок в формате с фиксированной точкой и пятью десятичными знаками после запятой.
Нажав клавишу F5, проверяем, как работает программа.
Результат работающей программы представлен на рис. 2.2.

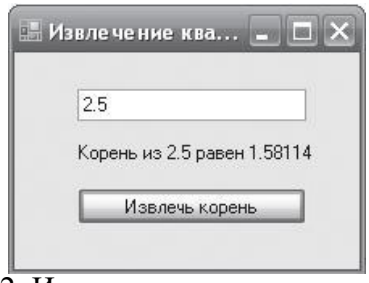


Рис.2.2. Извлечение квадратного корня

Контрольные вопросы

1. Опишите шаги, которые вы сделали, чтобы открыть диалоговую панель программы для ее визуальной настройки.

2. Что такое Class Wizard.
3. Как добавить элемент для ввода данных.

Итог работы: файлы, защита

Практическая работа №3.
«Анализ потоков данных». « Использование метрик стилистики»

Цель: Научиться разрабатывать и реализовывать программу, использующую вкладки и переключатели, изменять шрифт вкладок.
Задание. Разработать программу, позволяющую выбрать текст из двух вариантов, задать цвет и размер шрифта этого текста на трех вкладках TabControl с использованием переключателей RadioButton.

Программируя поставленную задачу, запустим Visual Studio и выберем приложение в среде CLR шаблона Windows Forms Application Visual C++. Назовем этот проект Вкладки. Используя панель элементов Toolbox, в форму перетащим с помощью мыши элемент управления TabControl. Как видно, по умолчанию имеем две вкладки, а по условию задачи, как показано на рисунке, три вкладки. Добавить третью вкладку можно в конструкторе формы, а можно программно (рис.3.1).

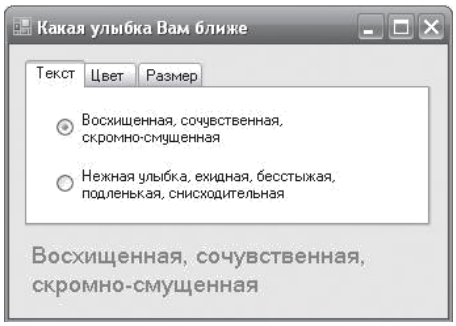


Рис.3.1 Программа с переключателями и вкладками

Чтобы добавить третью вкладку в конструкторе, необходимо в свойствах (окно Properties) элемента управления tabControl1 выбрать свойство TabPages, в результате попадаем в диалоговое окно TabPage Collection Edit, где добавляем (кнопка Add) третью вкладку (первые две присутствуют по умолчанию). Эти вкладки нумеруются от нуля, то есть третья вкладка будет распознаваться как TabPages(2). Название каждой вкладки будем указывать в программном коде.

Рассмотрим, как добавить третью вкладку не в конструкторе, а в программном коде при обработке события загрузки формы (листинг). Однако прежде чем перейти на вкладку программного кода, для каждой вкладки выбираем из панели Toolbox по два переключателя RadioButton, а в форму перетаскиваем метку Label. Теперь с помощью щелчка правой кнопкой мыши в пределах формы переключаемся на редактирование программного кода.

Листинг Фрагмент программы, управляющей вкладками и переключателями

```
//  .............
//  Программный код, расположенный выше, создан средой Visual
Studio
//  автоматически, поэтому автором не приводится
this->ResumeLayout(false);
this->PerformLayout();
```

```
}
#pragma endregion
//      Программа, позволяющая выбрать текст из двух вариантов, задать цвет
//     и размер шрифта для этого текста на трех вкладках TabControl
//     с использованием переключателей RadioButton
private: System::
Void Form1_Load(System::Object^ sender, System::EventArgs^ e) {
//     Создание третьей вкладки "программно":
auto tabPage3 = gcnew System::Windows::Forms::TabPage(); tabPage3-
>UseVisualStyleBackColor = true;
//     Добавление третьей вкладки в существующий набор
//     вкладок tabControl1:
this->tabControl1->Controls->Add(tabPage3);
//     Добавление переключателей 5 и 6 на третью вкладку: tabPage3-
>Controls->Add(this->radioButton5); tabPage3->Controls->Add(this-
>radioButton6);
//     Расположение переключателей 5 и 6:
this->radioButton5->Location = System::Drawing::Point(20, 15); this->radioButton6-
>Location = System::Drawing::Point(20, 58); this->Text = "Какая улыбка вам ближе";
//     Задаем названия вкладок:
tabControl1->TabPages[0]->Text = "Текст"; tabControl1-
>TabPages[1]->Text = "Цвет"; tabControl1->TabPages[2]-
>Text = "Размер";
//     Эта пара переключателей изменяет текст:
radioButton1->Text =
"Восхищенная, сочувственная,\nскромно-смущенная"; radioButton2-
>Text = "Нежная улыбка, ехидная, бес" + "стыжая,\nподленькая,
снисходительная";
//     или
//     radioButton2->Text = "Нежная улыбка, бесстыжая," +
//     Environment::NewLine + "подленькая, снисходительная";
//     Эта пара переключателей изменяет цвет текста: radioButton3-
>Text = "Красный"; radioButton4->Text = "Синий";
//     Эта пара переключателей изменяет размет шрифта: radioButton5-
>Text = "11 пунктов"; radioButton6->Text = "13 пунктов"; label1->Text
= radioButton1->Text;
}
private: System::Void radioButton1_CheckedChanged(System::Object^ sender,
System::EventArgs^ e)
{ label1->Text = radioButton1->Text; }
private: System::Void radioButton2_CheckedChanged(System::Object^ sender,
System::EventArgs^ e)
{ label1->Text = radioButton2->Text; }
private: System::Void radioButton3_CheckedChanged(System::Object^ sender,
System::EventArgs^ e)
{ label1->ForeColor = Color::Red; }
private: System::Void radioButton4_CheckedChanged(System::Object^ sender,
System::EventArgs^ e)
{ label1->ForeColor = Color::Blue; }
private: System::Void radioButton5_CheckedChanged(System::Object^ sender,
System::EventArgs^ e)
{ label1->Font = gcnew System::Drawing:: Font(label1-
>Font->Name, 11); }
```

```
private: System::Void radioButton6_CheckedChanged(System::Object^ sender,
System::EventArgs^ e)
{ label1->Font = gcnew System::Drawing:: Font(label1-
>Font->Name, 13); }
};
}
```

Как видно из текста программы, при обработке события загрузки формы Form1_Load (этот участок программного кода можно было бы задать сразу после вызова процедуры InitializeComponent) создаем «программно» третью вкладку. Заметьте, что мы ее объявили как auto, то есть тип переменной tabPage3 выводится из выражения инициализации в Visual C++. Далее добавляем новую вкладку tabPage3 в набор вкладок tabControl1, созданный в конструкторе. Затем «привязываем» пятый и шестой переключатели к третьей вкладке.

Каждая пара переключателей, расположенных на каком-либо элементе управления (в данном случае на различных вкладках), «отрицают» друг друга, то есть если пользователь выбрал один, то другой переходит в противоположное состояние. Отслеживать изменения состояния переключателей удобно с помощью обработки событий переключателей CheckChanged (см. листинг). Чтобы получить пустой обработчик этого события в конструкторе формы, следует дважды щелкнуть на соответствующем переключателе и таким образом запрограммировать изменения состояния переключателей.

Контрольные вопросы

1. Опишите, как добавить в программу вкладки.
2. По какому принципу работают переключатели RadioButton.
3. Опишите, как организовать работу группы переключателей.

Итог работы: файлы, ответы на контрольные вопросы, отчет

Практическая работа №4.

«Выполнение измерений характеристик кода в среде Visual Studio».

Цель: Научиться программировать консольное приложение, для вычисления математических функций, вводимых значений и вывода результата на экран. Получить практические навыки по использованию различных элементов графического интерфейса и операторов языка VC++.

Теоретические сведения

Иногда, например для научных расчетов, требуется организовать какой-нибудь самый простой ввод данных, выполнить весьма сложную математическую обработку введенных данных и оперативно вывести на экран результат вычислений.

Можно по-разному организовать такую программу, в том числе программируя так называемое *консольное приложение* (от англ. *console* — пульт управления). Под консолью обычно подразумевают экран компьютера и клавиатуру.

Задание: Напишем консольное приложение, которое приглашает пользователя ввести два числа , складывает их и выводит результат вычислений на консоль. Для этого запускаем Visual C++ 2010, далее создаем новый проект (New Project), в узле Visual C++ в среде CLR выбираем шаблон Console Application CLR, задаем имя решения (Name) — Сумма. После щелчка на кнопке OK попадаем сразу на вкладку программного кода (рис. 4.1).

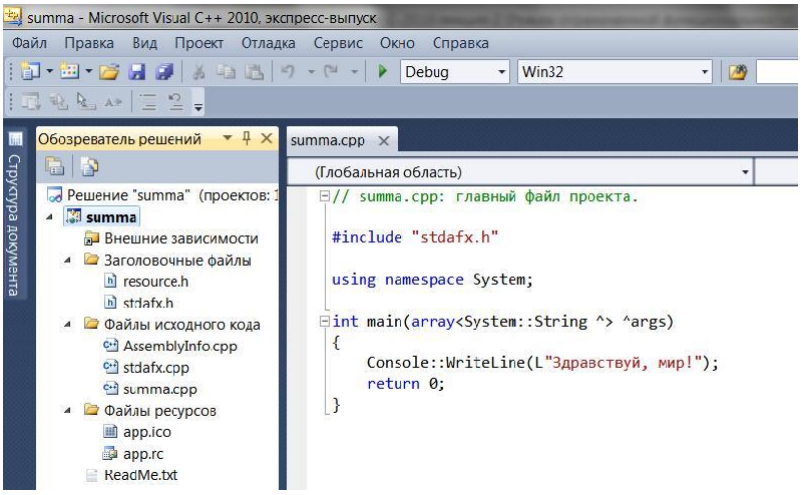


Рис. 4.1. Вкладка программного кода

Как видите, здесь управляющая среда Visual Studio приготовила несколько строк программного кода. Это вполне работоспособная программа. При запуске консольного или Windows-приложения C++ метод Main() является первым вызываемым методом. В фигурные скобки после Main() мы вставим собственный программный код (листинг). Фрагмент работы программы на рисунке 4.2.

Листинг Ввод и вывод данных в консольном приложении

```
//     Сумма.cpp: главный файл проекта.
//      Программа организует ввод двух чисел, их сложение и вывод суммы на консоль

#include "stdafx.h"
using namespace System;
int main(array<System::String ^> ^args)
{
//     Задаем строку заголовка консоли: Console::Title = "Складываю два числа:";
Console::BackgroundColor = ConsoleColor::Cyan; // - цвет фона Console::ForegroundColor =
ConsoleColor::Black; // - цвет текста Console::Clear();

//     Ввод первого слагаемого:
Console::WriteLine("Введите первое слагаемое:"); String^ Строка =
Console::ReadLine(); Single X, Y, Z;
//     Преобразование строковой переменной в число: X =
Single::Parse(Строка);
//     Ввод второго слагаемого: Console::WriteLine("Введите второе
слагаемое:");
Строка = Console::ReadLine(); Y =
Single::Parse(Строка);
Z=X+Y;
Console::WriteLine("Сумма = {0} + {1} = {2}", X, Y, Z);
//     Звуковой сигнал частотой 1000 Гц и длительностью 0.5 секунды:
Console::Beep(1000, 500);
//      Приостановить выполнение программы до нажатия какой-нибудь клавиши:
Console::ReadKey(); return
0;
}
```

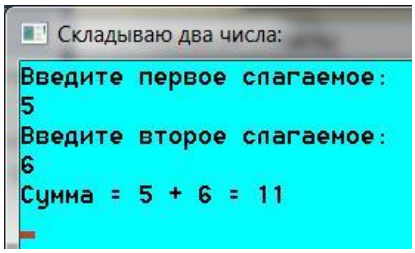


Рис. 4.2. Фрагмент работы консольного приложения

Итак, в данной программе Main() — это стартовая точка, с которой начинается ее выполнение. Обычно консольное приложение выполняется в окне на черном фоне. Чтобы как-то украсить традиционно черное окно консольного приложения, установим цвет фона окна BackgroundColor сине-зеленым (Cyan), а цвет символов, выводимых на консоль, черным (Black). Выводим строки в окно консоли методом WriteLine, а для считывания строки символов, вводимых пользователем, используем метод ReadLine. Далее объявляем три переменных типа Single для соответственно первого числа, второго и значения суммы. Тип данных Single применяется тогда, когда число, записанное в переменную, может иметь целую и дробную части.

Переменная типа Single занимает 4 байта. Для преобразования строки символов, введенных пользователем в числовое значение, используем метод Parse.

После вычисления суммы необходимо вывести результат вычислений из оперативной памяти на экран. Для этого воспользуемся форматированным выводом в фигурных скобках метода WriteLine объекта Console:

Console::WriteLine("Сумма = {0} + {1} = {2}", X, Y, Z)

Затем выдаем звуковой сигнал Beep, символизирующий об окончании процедуры и выводе на экран результатов вычислений. Последняя строка в программе Console::ReadKey(); предназначена для приостановки выполнения программы до нажатия какой-нибудь клавиши. Если не добавить эту строку, окно с командной строкой сразу исчезнет, и пользователь не сможет увидеть вывод результатов выполнения. Программа написана. Нажмите клавишу F5, чтобы увидеть результат.

Контрольные вопросы

1. Что такое форматированный ввод?
2. Какая команда помогает задерживать результат работы программы на экране?
3. Опишите вывод результата.

Итог работы: файлы, защита

Практическая работа №5.

«Выполнение измерений характеристик кода в среде (например, Eclipse C/C++ и др.)»

Цель: Научиться создавать элементы управления в форме «программным» способом, обрабатывать несколько событий одной процедурой. Получить практические навыки по использованию различных элементов графического интерфейса и операторов языка VC++.

Задание:

1. Создать новый проект с формой. При этом, как обычно, запускаем Visual Studio 2010, в окне New Project выбираем в среде

CLR узла Visual C++ приложение шаблона Windows Forms Application Visual C++. Чтобы к программному коду добавить пустой обработчик события загрузки формы, дважды щелкнем на

проектируемой экранной форме. Далее вводим программный код, представленный в листинге :

```
// ..............
// Программный код, расположенный выше, создан средой Visual
Studio
// автоматически, поэтому автором не приводится
this->Load += gcnew System::EventHandler(this,
&Form1::Form1_Load);
this->ResumeLayout(false);
}
#pragma endregion
// Программа создает командную кнопку в форме «программным»
способом,
// т.е. с помощью написания непосредственно программного кода,
не
// используя при этом панель элементов управления Toolbox.
Программа
// задает свойства кнопки: ее видимость, размеры, положение,
надпись
// на кнопке и подключает событие "щелчок на кнопке"
private: System::Void Form1_Load(System::Object^ sender,
System::EventArgs^ e)
{
// Создание кнопки без панели элементов управления:
Button^ button1 = gcnew Button();
// Задаем свойства кнопки:
button1->Visible = true;
// Ширина и высота кнопки:
button1->Size = Drawing::Size(100, 30);
// Расположение кнопки в системе координат формы:
button1->Location = Drawing::Point(100, 80);
button1->Text = "Новая кнопка";
// Добавление кнопки в коллекцию элементов управления
this->Controls->Add(button1);
// Подписку на событие Click для кнопки можно делать "вручную".
// Связываем событие Click с процедурой обработки этого события:
button1->Click += gcnew EventHandler(this,
&Form1::ЩелчокНаКнопке);
}
private: System::
Void ЩелчокНаКнопке(System::Object^ sender, System::EventArgs^
e)
{
MessageBox::Show("Нажата новая кнопка");
}
};}
```

Мы видим, что при обработке события загрузки формы создаем новый объект button1 стандартного класса кнопок. Задаем свойства кнопки: ее видимость (Visible), размеры (Size), положение (Location) относительно левого нижнего угла формы, надпись на кнопке — ≪Новая кнопка≫.

Далее необходимо организовать корректную работу с событием «щелчок на созданной нами командной кнопке». В предыдущих примерах мы для этой цели в

конструкторе формы дважды щелкали на проектируемой кнопке, и исполняемая среда автоматически генерировала пустой обработчик этого события в программном коде. Или опять же в конструкторе формы в панели свойств проектируемой кнопки щелкали мышью на значке молнии (Events) и в появившемся списке всех событий выбирали необходимое событие. Однако согласно условию задачи мы должны организовать обработку события «программным» способом без использования конструктора формы. Для этого в программном коде сразу после добавления командной кнопки в коллекцию элементов управления поставим оператор стрелки (->) после имени кнопки button1 и в раскрывающемся списке выберем необходимое событие Click. Затем, как приведено в листинге 3.2, осуществляем так называемую «подписку» на данное событие, то есть с помощью ключевого слова EventHandler связываем событие Click с процедурой обработки события. Мы его назвали ≪Щелчок-НаКнопке≫. Теперь создадим обработчик события «щелчок на кнопке», как показано в листинге. В этой процедуре предусматриваем вывод сообщения ≪ Нажата новая кнопка≫. На рисунке 5.1 приведен фрагмент работы программы.

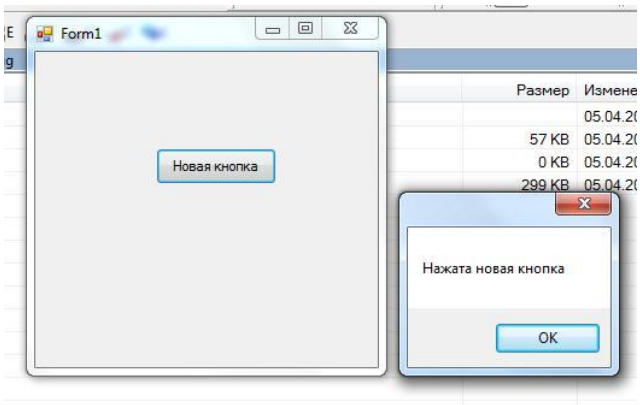


Рис.5.1 Фрагмент работы программы

В заключение отметим, что в случае создания пустого обработчика события в конструкторе формы строка подписки на событие формируется автоматически в методе InitializeComponent в файле Form1.h проекта.

Запустить Visual Studio 2010, в окне New Project выберем в среде CLR узла Visual C++ приложение шаблона Windows Forms Application Visual C++. Затем из панели элементов перенесем в форму две командные кнопки и текстовую метку. Далее через двойной щелчок мышью в пределах проектируемой формы создать пустой обработчик загрузки формы и перейдем к вкладке программного кода (листинг).

```
// .............
// Программный код, расположенный выше, создан средой Visual
Studio
// автоматически, поэтому автором не
приводится this->ResumeLayout(false); this-
>PerformLayout();
}
#pragma endregion
```

```
// В форме имеем две командные кнопки, и при нажатии указателем
мыши
//  любой из них получаем номер нажатой кнопки. При этом в
программе
// предусмотрена только одна процедура обработки событий
private: System::Void Form1_Load(System::Object^ sender,
System::EventArgs^ e)
{
Form1::Text = "Щелкните на кнопке"; label1->Text = nullptr;
// Связываем события Click от обеих кнопок с одной
процедурой КЛИК:
button1->Click += gcnew EventHandler(this, &Form1::КЛИК);
button2->Click += gcnew EventHandler(this, &Form1::КЛИК);
// Подпиской на событие называют связывание названия события
// с названием процедуры обработки события посредством
EventHandler
}
private: System::Void КЛИК(System::Object^ sender,
System::EventArgs^ e)
{
// String S = Convert.ToString(sender);
// получить текст, отображаемый на кнопке, можно таким образом:
Button^ Кнопка = (Button^)sender;
// или String^ НадписьНаКнопке = ((Button^)sender)->Text;
label1->Text = "Нажата кнопка " + Кнопка->Text; // или
Кнопка->Name
}
};}
```

Как видно из текста программы, при обработке события загрузки формы мы осуществляем так называемую подписку на событие, то есть связываем название события с названием процедуры обработки события КЛИК посредством метода (делегата) EventHandler. Этот метод делегирует (передает полномочия) обработку события button1->Click процедуре КЛИК. Заметим, что события Click от обеих кнопок мы связали с одной и той же процедурой КЛИК.

Далее создаем процедуру обработки события КЛИК, ее параметр sender содержит *ссылку на объект-источник события*, то есть кнопку, нажатую пользователем.
С помощью неявного преобразования можно конвертировать параметр sender в экземпляр класса Button и, таким образом, выяснить все свойства кнопки, которая инициировала событие. На рисунке 5.2 приведен пример работы написанной программы.

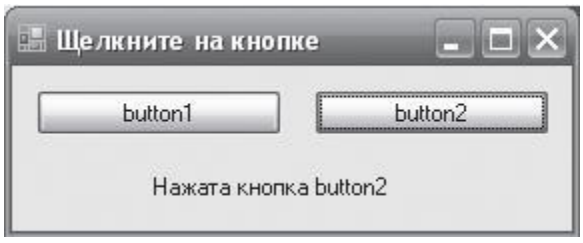


Рис.5.2 Фрагмент работы программы, определяющей нажатую кнопку

Контрольные вопросы

1. Опишите создание кнопок Button.
2. Опишите свойства Size, Point.
3. Что такое обработчик событий?
4. Опишите процедуру gcnew EventHandler.

5. Для чего нужен параметр sender?
6. Опишите связывание событий.

Итог работы: файлы, ответы на вопросы

Практическая работа №6.

«Определение проекта и его границ».
«Организационные структуры». «Методологии ведения проектов».

Цель: Научиться создавать приложение для обработки событий клавиатуры. Получить практические навыки по использованию различных элементов графического интерфейса и операторов языка VC++.

Задание: Выполните последовательность действий:

Напишите программу, информирующую пользователя о тех клавишах и комбинациях клавиш, которые тот нажал. Запустим Visual Studio 2010, в окне New Project выберем в среде CLR узла Visual C++ приложение шаблона Windows Forms Application Visual C++. Затем из панели Toolbox перетащим в форму две текстовых метки Label.

Далее, поскольку нам потребуются клавишные события формы: KeyPress, KeyDown, KeyUp, уже привычным способом получим пустые обработчики этих событий. То есть в панели Properties щелкнем на пиктограмме молнии (Events), а затем в списке всех возможных событий выберем каждое из названных событий клавиатуры.
Программный код приведен **в листинге** :

```
// ..............
//  Программный код, расположенный выше, создан средой Visual
Studio автоматически
this->ResumeLayout(false);
this->PerformLayout();
}
#pragma endregion
// Программа, информирующая пользователя о тех клавишах
// и комбинациях клавиш, которые тот нажал
private: System::
Void Form1_Load(System::Object^ sender, System::EventArgs^
e) {
// Устанавливаем шрифт с фиксированной шириной (моноширинный):
Form1::Font = gcnew Drawing::
Font(FontFamily::GenericMonospace, 14.0F);
// Поскольку мы задали этот шрифт увеличенным (от 8 по умолчанию
// до 14), форма окажется пропорционально увеличенной
Form1::Text = "Какие клавиши нажаты сейчас:"; label1->Text
= String::Empty; label2->Text = String::Empty;
}
private: System::Void Form1_KeyPress(System::Object^
sender, System::Windows::Forms::KeyPressEventArgs^ e)
{
// Здесь событие нажатия клавиши: при удержании
// клавиши генерируется непрерывно
label1->Text = "Нажатая клавиша: " + e->KeyChar;
}
private: System::Void Form1_KeyDown(System::Object^ sender,
System::Windows::Forms::KeyEventArgs^ e)
{
// Здесь обрабатываем мгновенное событие первоначального
```

```
// нажатия клавиши
label2->Text = String::Empty;
// Если нажата клавиша Alt
if (e->Alt == true) label2->Text += "Alt: Yes\n";
else label2->Text += "Alt: No\n";
// Если нажата клавиша Shift
if (e->Shift == true) label2->Text += "Shift: Yes\n";
else label2->Text += "Shift: No\n";
// Если нажата клавиша Ctrl
if (e->Control == true) label2->Text += "Ctrl: Yes\n";
```

26

```
else label2->Text += "Ctrl: No\n";
label2->Text += String::Format(
"Код клавиши: {0} \nKeyData: {1} \nKeyValue:
{2}", e->KeyCode, e->KeyData, e->KeyValue);
}
private: System::Void Form1_KeyUp(System::Object^
sender, System::Windows::Forms::KeyEventArgs^ e) {

// Очистка меток при освобождении клавиши
label1->Text = String::Empty; label2->Text = String::Empty;
}
};
}
```

В первую метку label1 записываем сведения о нажатой обычной (то есть не модифицирующей и не функциональной) клавише при обработке события KeyPress.

Во вторую метку из аргумента события e (e->Alt, e->Shift и e->Control) получаем сведения, была ли нажата какая-либо модифицирующая клавиша (либо их комбинация). Обработчик события KeyUp очищает обе метки при освобождении клавиш.
На рисунке 6.1 приведен фрагмент работы программы.

25

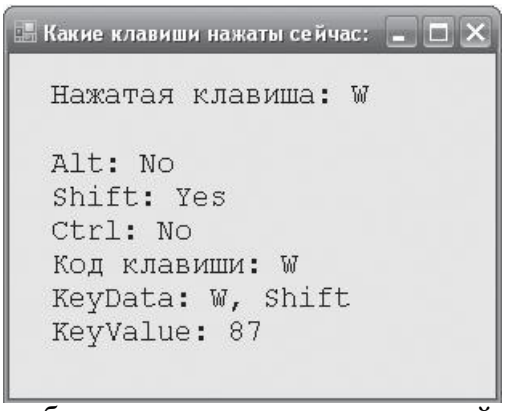


Рис. 6.1 Фрагмент работы программы, определяющей нажатую клавишу

Контрольные вопросы

1. Что означает модифицирующая клавиша?
2. Какое событие необходимо обработать, чтобы узнать нажата ли модифицирующая клавиша?
3. В какой момент нажатия клавиши генерируется событие KeyDown, KeyUp?

Итог работы: файлы, ответы на вопросы

Практическая работа №7.

«Построение команды проекта». «Начало и завершение проекта»

Цель: Изучить приемы работы с файлами и способы создания файлов. Получить практические навыки по использованию файлов для хранения информации.

Задание:

Напишите программу, содержащую на экранной форме текстовое поле и две командные кнопки. При щелчке мышью на первой кнопке происходит чтение текстового файла в текстовое поле в кодировке Unicode. При щелчке на второй кнопке отредактированный пользователем текст в текстовом поле сохраняется в файл на диске.

Запустим систему Visual Studio 2010 и в окне New Project выберем
в среде CLR узла Visual C++ приложение шаблона Windows Forms Application Visual C++. Далее в форму из панели Toolbox перенесем текстовое поле и две командные копки. Для текстового поля в окне Properties сразу укажем для свойства Multiline значение True, чтобы текстовое поле имело не одну строку, а столько, сколько поместится в растянутом указателем мыши поле. Одна кнопка предназначена для открытия файла, а другая — для сохранения файла на машинном носителе. В листинге приведен текст данной программы «Чтение/запись текстового файла в кодировке Unicode».

Листинг:

```
// ..............
// Программный код, расположенный выше, создан средой Visual
Studio
// автоматически, поэтому автором не приводится
this->ResumeLayout(false);
this->PerformLayout();
}
#pragma endregion
// Программа для чтения/записи текстового файла в кодировке
Unicode
String ^ filename;
// Объявляем filename здесь, чтобы эта переменная была "видна"
// в процедурах обработки обоих событий.
private: System::Void Form1_Load(System::Object^
sender, System::EventArgs^ e)
{
// Установка начальных значений:
textBox1->Multiline = true; textBox1->Clear();
textBox1->Size = Drawing::Size(268, 112);
button1->Text = "Открыть"; button1->TabIndex = 0;
button2->Text = "Сохранить";

Form1::Text = "Здесь кодировка
Unicode"; filename = "C:\\Text1.txt";
}
private: System::Void button1_Click(System::Object^ sender,
System::EventArgs^ e)
{
// Щелчок на кнопке Открыть.
// Русские буквы будут корректно читаться,
// если открыть файл в кодировке UNICODE:
try
{
```

```
// Создание объекта StreamReader для чтения из файла:
auto Читатель = gcnew IO::StreamReader(filename);
// Непосредственное чтение всего файла в текстовое
поле: textBox1->Text = Читатель->ReadToEnd();
Читатель->Close(); // закрытие файла
// Читать текстовый файл в кодировке UNICODE в массив строк
// можно также таким образом (без Open и Close):
// array <String^>^ МассивСтрок =
// IO::File::ReadAllLines("C:\\Text1.txt");
}
catch (IO::FileNotFoundException^ Ситуация)
{ // Обработка исключительной ситуации:
MessageBox::Show(Ситуация->Message + «\nНет такого файла»,
"Ошибка", MessageBoxButtons::OK,
MessageBoxIcon::Exclamation);
}
catch (Exception^ Ситуация)
{
// Отчет о других ошибках:
MessageBox::Show(Ситуация->Message, "Ошибка",
MessageBoxButtons::OK,
MessageBoxIcon::Exclamation); }
}
private: System::Void button2_Click(System::Object^ sender,
System::EventArgs^ e)
{
// Щелчок на кнопке Сохранить:
try
{
// Создание объекта StreamWriter для записи в
файл: auto Писатель = gcnew
IO::StreamWriter(filename, false); Писатель-
>Write(textBox1->Text); Писатель->Close();

// Сохранить текстовый файл можно также таким образом
// (без Close), причем, если файл уже существует,
// то он будет заменен:
// IO::File::WriteAllText("C:\\tmp.tmp", textBox1->Text);
}
catch (Exception^ Ситуация)
{
// Отчет обо всех возможных ошибках:

MessageBox::Show(Ситуация->Message, "Ошибка",
MessageBoxButtons::OK, MessageBoxIcon::Exclamation);
}
}
};
}
```

Блоки try, которые, как мы видим, используются в данном программном коде. Логика использования try следующая: *попытаться* (try) выполнить некоторую задачу, например прочитать файл. Если задача решена некорректно (например, файл не найден), то *«перехватить »* (catch) управление и *обработать* возникшую (*исключительную*, Exception) ситуацию.

При обработке события «щелчок на кнопке Открыть» организован ввод файла C:\Text1.txt. Обычно в этой ситуации пользуются элементом управления OpenFileDialog для выбора файла. Мы не стали использовать этот элемент управления для того, чтобы *не «заговорить»* проблему, а также свести к минимуму программный код.

Далее создаем объект (поток) Читатель для чтения из файла. Для большей выразительности операций с данным объектом мы назвали его русскими буквами.

При обработке события «щелчок на кнопке Сохранить» организована запись файла на диск аналогично через объект Писатель. При создании объекта Писатель первым аргументом является filename, а второй аргумент false указывает, что данные следует *не добавить* (append) к содержимому файла (если он уже существует), а *перезаписать* (overwrite). Запись на диск производится с помощью метода Write() из свойства Text элемента управления textBox1. На рисунке 7.1 приведен фрагмент работы программы.

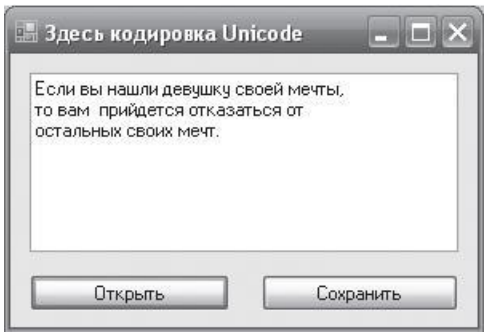


Рис.7.1 Чтение/запись текстового файла в кодировке Unicode

Запись текстового файла с помощью данной программы будет происходить *в формате (кодировке) Unicode*, как и чтение из файла. То есть вы сможете читать эти файлы Блокнотом, редактировать их, но каждый раз при сохранении файлов следить, чтобы кодировка была (оставалась) Unicode.

Контрольные вопросы

К чему свелась обработка исключительной ситуации ?
К каким методом происходит чтение файла filename?
К каким методом происходит закрытие файла?

Итог работы: файл, ответы на контрольные вопросы

Практическая работа №8.

«Коммуникации в проекте». «Построение иерархической структуры работ проекта».

Цель: Научиться создавать меню, передавать значения между диалоговыми окнами и главным окном приложения. Получить практические навыки в разработке программ.

Задание:

1. Создайте приложение с помощью Visual Studio 2010, в окне New Project выберте в среде CLR узла Visual C++ приложение шаблона
Windows Forms Application
Для создания меню на панели инструментов выберите *MenuStrip* . Дважды кликните на появившемся в нижней области окна объекте,
а затем перейдите на форму и в области *(Вводить здесь)* введите меню верхнего уровня с текстом *Цвет*.
Переместитесь на нижнюю область и введите текст *Черный.* Заполните элемент MenuStrip следующим образом:

Цвет
Черный
Красный
Синий
Зеленый

Запустите программу и поэкспериментируйте: выбирайте разные элементы созданного объекта.
Запрограммируйте событие *Click* для каждого пункта; например, для элемента *Черный* необходимо написать следующий код (дважды щелкнув на пункте, чтобы открыть код):

```
private: System::Void черныйToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e) {
  this->txt->BackColor=System::Drawing::Color::Black;
}
```

Запрограммируйте аналогично остальные пункты *Цвет*. Запустите и отладьте приложение. Сохраните проект.

Контрольные вопросы

1. Каково основное назначение объекта MenuStrip?
2. Как запрограммировать необходимый пункт меню формы в Visual c++?
3. Какое свойство служит для изменения фона объекта?
4. С помощью какого свойства menu можно сделать недоступным какой-либо пункт?

Итог работы: файл, защита, ответы на вопросы.

Практическая работа №9.
«Смета затрат на разработку и реализацию проекта»

Цель: составить смету затрат на разработку и реализацию проекта. Изучить возможности программного продукта Microsoft Project, предназначенного для управления проектами

Задание1. Изучить теоретические сведения:

Разработка смет — процесс структуризации и систематизации стоимостных оценок, полученных на этапе оценки стоимости. Структуризация и систематизация данных о стоимости работ производится в соответствии со статьями затрат, принятыми в системе учета родительской организации проекта.

Смета — документ, содержащий список затрат проекта, полученных на основе объемов работ проекта, требуемых ресурсов и цен, структурированный по статьям.
Если в проекте (родительской организации) проектные сметы принято структурировать по работам, то процесс разработки смет значительно упрощается. Оценки, структурированные по работам, переносятся в смету и сводятся в единый документ.

Если же требованием компании является структуризация расходов в смете по статьям затрат, процесс несколько усложняется. Обычно выделяют:

- прямые затраты (расходы);
- накладные (косвенные) затраты;
- общие и административные накладные расходы.

Прямые затраты — расходы, непосредственно связанные с производством продукции, работ проекта; производственные расходы, включаемые в себестоимость продукции, в прямые издержки производства.

Прямые расходы напрямую связаны с пакетом работ. Они включают:

- затраты на оплату труда;
- затраты на материалы и оборудование;
- иные расходы, связанные с выполнением работ.

Именно на прямые расходы могут непосредственно влиять менеджер проекта и его команда. Влияние команды проекта на другие расходы ограниченно.

Накладные расходы (косвенные затраты) — расходы, сопровождающие, сопутствующие основному производству, но не связанные с ним напрямую, не входящие в стоимость труда и материалов. Накладные расходы не могут быть привязаны к какой-то конкретной работе, конкретному результату. Они относятся ко всему проекту в целом. Это затраты на:

- содержание и эксплуатацию основных средств;
- управление, организацию, обслуживание производства;
- командировки;
- обучение работников.

Общие и административные накладные расходы (постоянные расходы) — затраты, не связанные с каким-то конкретным проектом. Они относятся к расходам компании, но при этом имеют отношение и к проекту. К общим и административным расходам обычно относятся расходы на содержание аппарата управления, поддерживающих подразделений (бухгалтерия, секретариат, охрана и др.).

Задание: Оформите смету затрат на разработку и реализацию проекта.

Таблица 1 Расчет материальных затрат.

Наименование	
Единица измерения	
Количество	
Стоимость, руб.	
за единицу, руб.	
общая, руб.	
ИТОГО:	
Транспортные расходы (15%):	
ВСЕГО:	

Таблица 2 Смета выполненных работ.

Наименование вида выполненных работ	
Единица измерения	
Количество	
Стоимость, руб. выполненные работы в том числе З/П	
за единицу, руб.	
общая, руб.	
ИТОГО:	
Наименование	
Единица измерения	
Количество	
Стоимость, руб.	
за единицу, руб.	
общая, руб.	
Транспортные расходы (15%):	
Всего с транспортными расходами:	
Плановые накопления (4-8% от итого материалы и выполненные работы)	
Всего с плановыми накоплениями	

Сводка итогов:	
А) Материальные затраты	
Б) Выполненные работы и материалы	
Всего:	
Заработная плата:	
Начисления на заработную плату:	
Всего с начислениями:	
ВСЕГО ПО СМЕТЕ	

Задания 2:

1. Познакомиться с режимами и основными функциями программы Microsoft Project.
2. Изучить возможности создания и оптимизации проектов с помощью программы Microsoft Project.
3. Выполнить индивидуальное задание.
4. Подготовить отчет о проделанной работе.

Порядок выполнения работы

1. Основные сведения о программе

Для поддержки процесса управления проектом на различных этапах существует большое количество программных комплексов, целью применения которых является повышение эффективности реализации проекта, т. е. выполнение как всего проекта в целом, так и его отдельных этапов в заданные сроки и в рамках утвержденных денежных ресурсов. Основными задачами менеджера проекта является составление сетевого графика (расписания) проекта, распределение ресурсов между задачами проекта и слежение за ходом реализации проекта.

Внедрение компьютерных технологий в практику реализации проекта может оказать существенную помощь в эффективной реализации проектов.

Анализ существующих программных комплексов поддержки деятельности по управлению проектами показал, что для малых проектов, в которых удельный вес этапа реализации значителен, предпочтительным является пакет программ Microsoft Project (MS Project). Он позволяет эффективно выполнить структуризацию проекта путем разделения его на этапы и подзадачи, выявить критические задачи (задачи, длительность которых существенно влияет на длительность реализации всего проекта), получить сетевой график проекта, осуществить назначение ресурсов задачам проекта, контролировать загрузку ресурсов.

Microsoft Project позволяет эффективно управлять проектом на различных этапах его реализации. Он дает возможность выполнить структуризацию проекта путем разделения его на этапы, задачи и подзадачи, выявить критические задачи (задачи, длительность которых существенно влияет на длительность реализации всего проекта), получить сетевой график и календарный план проекта, осуществить назначение ресурсов задачам проекта, эффективно контролировать загрузку ресурсов. Пакет поддерживает все необходимые типы связей между задачами: FS (Finish-Start), SS (Start-Start), FF (Finish-Finish).

Поддерживая современные информационные технологии, пакет MS Project позволяет импортировать данные из файлов, созданных в среде других приложений, например MS Excel и MS Access. Неоспоримым достоинством пакета является наличие встроенного языка программирования Visual Basic For Application, что обеспечивает возможность разработки программных компонент, обеспечивающих решение специфических задач.

2. Построение модели

Методика использования пакета Microsoft Project для управления инновационным проектом на этапе подготовки к реализации, целью которой является получение сетевого графика и календарного плана проекта, может быть представлена в виде последовательности следующих шагов:

- создание календаря проекта (т. е. учет нерабочих и праздничных дней);
- составление списка задач, которые надо выполнить для успешной реализации проекта;
- определение связей между задачами;
- выявление задач, длительность реализации которых существенно влияет на длительность реализации всего проекта, и, возможно, изменение порядка выполнения задач проекта;
- формирование списка доступных для реализации проекта ресурсов;
- распределение ресурсов (назначение ресурсов конкретным задачам проекта).

Начало работы над проектом

Календарь

Одной из задач, решаемых на этапе подготовки к реализации проекта, является получение сетевого графика проекта. При построении сетевого графика MS Project использует календарь — таблицу, в которой отражены рабочие и нерабочие (выходные и праздничные) дни. Различают стандартный календарь и календари пользователя. Стандартный календарь предполагает, что рабочими днями являются все дни недели, за исключением субботы и воскресенья, и рабочий день длится 8 часов. Календарь пользователя учитывает реальные рабочие дни. Например, очевидно, что в России реальный календарь в январе и мае существенно отличается от стандартного.

При работе с проектом MS Project позволяет, помимо базового календаря, использовать несколько календарей пользователя. Можно, например, создать календарь для всего проекта в целом (основной календарь) и календарь для отдельного ресурса, который будет учитывать особенности его использования (например, недоступность ресурса в определенные дни или месяцы).

Задачи проекта

Задача – это некоторая деятельность, которую надо выполнить, чтобы завершить часть проекта. Работа над проектом начинается с составления списка задач проекта.

Большие, сложные задачи, как правило, могут быть естественным образом представлены в виде набора боле простых, более конкретных задач. Поэтому при составлении списка сначала записывают общую (обобщенную) задачу, затем – задачи, из которых эта общая задача состоит (подчиненные задачи).

При составлении списка задач проекта обычно используют метод, который часто называют «сверху — вниз». Суть метода заключается в том, что сначала составляют список главных (обобщенных) задач, затем этот список уточняется посредством добавления уточняющих задач.

Каждая задача проекта характеризуется длительностью, которая измеряется в минутах, часах, днях и неделях. Длительность обобщенной задачи определяется длительностью ее подчиненных задач. Длительность обобщенной задачи вычисляет MS Project.

Длительность подчиненной задачи нижнего уровня, т. е. задачи? у которой нет подчиненных задач, определяется временем необходимым для ее выполнения одной единицей ресурса. Например, один рабочий копает траншею в 5 метров 8 часов. Если для реализации проекта необходимо выкопать траншею длинной в 10 метров, то длительность задачи «Траншея» равна 16 часам. Длительность подчиненной задачи задает менеджер проекта на основе нормативной документации.

Контрольные точки проекта

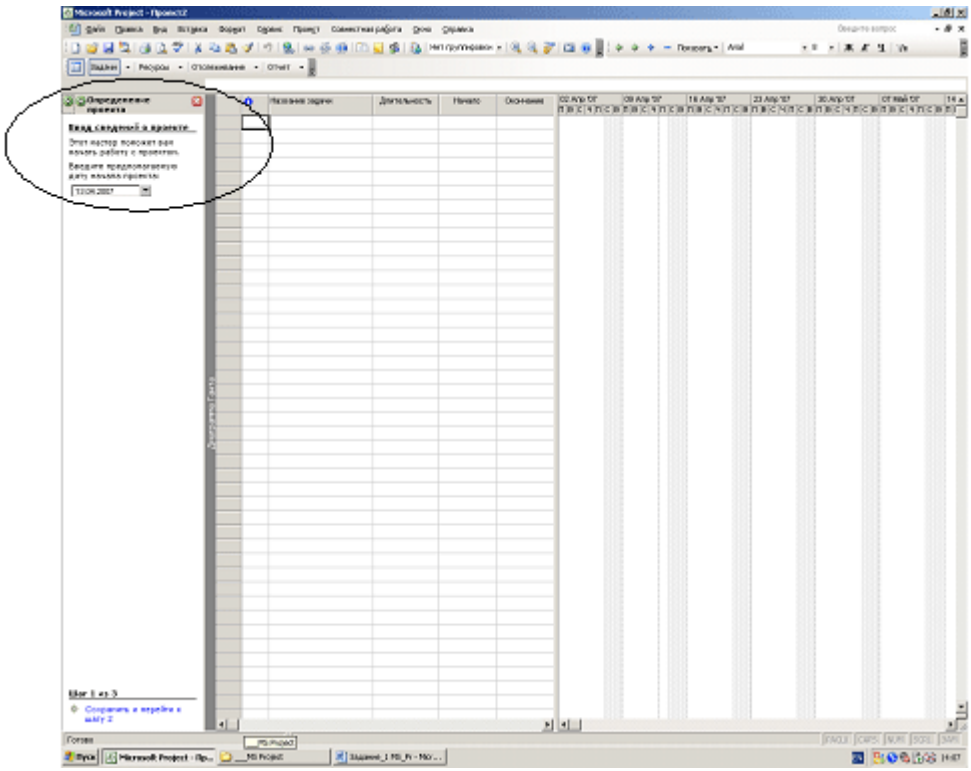
В каждом проекте могут быть выделены этапы, после выполнения которых процесс реализации проекта переходит на новый качественный уровень. Например, после этапа согласования требований с заказчиком начинается этап подготовки технической документации. Можно считать, что последней задачей этапа согласования требований с заказчиком является задача утверждения требований. Подобные задачи, как правило, должны выполняться в конкретный, заранее установленный день, поэтому их называют контрольными точками. Длительность задач – контрольных точек – принимают равной нулю.

Задание 1:

1. Запустите Microsoft Project.
2. Создайте календарь проекта.
3. Установите предполагаемую дату начала реализации проекта, задайте название проекта.
4. Используя таблицу ввода списка задач, введите названия задач проекта. Для подчиненных задач нижнего уровня задайте длительность. (***Замечание*:** На первом этапе работы над проектом, когда задачи проекта представлены в виде простого списка, длительность обобщенной задачи равна длительности самой длительной подчиненной задачи. Позже, когда будут установлены связи между задачами, длительность обобщенной задачи будет вычислена как сумма длительностей подчиненных задач.)
5. Если выполнение какой-либо задачи должно начаться в определенный день, то введите дату начала выполнения этой задачи.
6. Сохраните проект в своем рабочем каталоге.

Пример выполнения

Данные о проекте в MS Project 2003 вводятся с помощью Мастера.



Определение опорных дат проекта

Опорными данными для каждого проекта являются даты начала и окончания проекта. Одна из них (любая) вводится разработчиком; другая в дальнейшем вычисляется автоматически. ***Установить начальную дату проекта: 19 апреля.***

Определение общих рабочих часов для проекта (Создание календаря)

Project содержит несколько шаблонов календаря, которые можно использовать как основу для календаря проекта.

Используя подсказки Мастера, установить:

Рабочие дни: с понедельника по субботу.

Время работы: с 8.30 до 17.00; в субботу — с 8.30 до 15.00.

Задать праздничные и выходные дни.

Проанализировать изменения графика рабочего времени.

Дополнительные календари (они связаны с ресурсами проекта) пока не определять.

Ввод задач проекта

Перечень задач может быть введен непосредственно в программе MS Project или перенесен из других программ (Копировать/Вставить).

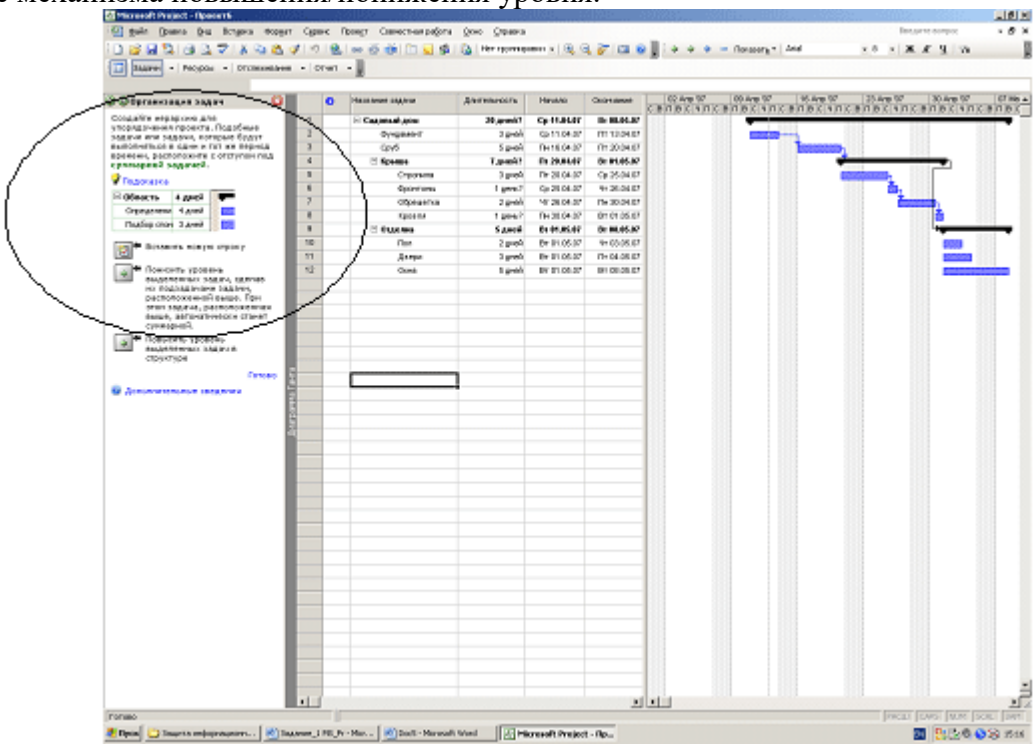
Ввести следующие задачи (проект строительства дома):

- Садовый дом
    - Фундамент
    - Сруб
    - Крыша
        - Стропила
        - Фронтоны
        - Обрешетка
        - Кровля
    - Отделка
        - Пол
        - Двери
        - Окна

Эти задачи в MS Project вводятся сначала простым списком, без указания их подчиненности. Указать длительность каждой задачи (произвольно).

Организация этапов задач (Задание иерархии задач)

Задачи проекта, как правило, не бывают равноправными. Список задач представляет собой некоторую иерархическую структуру. В MS Project такая структура реализуется на основе механизма повышения/понижения уровня.

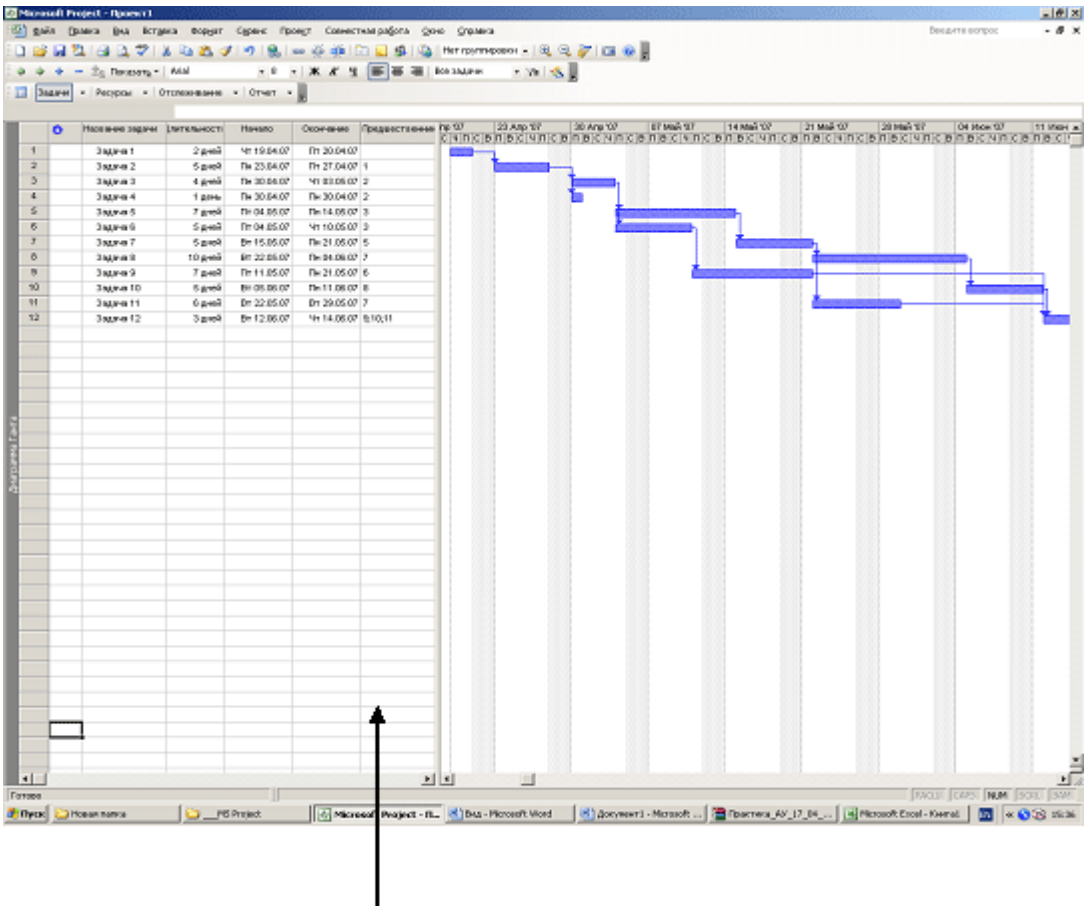


Планирование задач (Задание связей между задачами)

Задачи реального проекта связаны между собой во времени. Какие-то задачи могут выполняться одновременно (связь SS «начало–начало», например, ***Отделка дверей*** и ***Отделка окон***); другие не могут быть начаты, пока не закончатся некоторые предыдущие задачи (связь FS «конец-начало», например, ***Закладка фундамента*** и ***Создание сруба***); третьи должны заканчиваться одновременно (связь FF «конец-конец»). Поэтому после ввода списка задач и создания иерархии необходимо установить связи между задачами.

Связь SS устанавливается кнопкой ***«Связать задачи»*** на Панели инструментов. (Рядом находится кнопка «Разорвать связи»). Применить это к задачам ***Закладка фундамента*** и ***Создание сруба.***

Задание для самостоятельной работы: Открыть новый файл «Учебный проект». Ввести список задач, предварительно скопировав его из файла MS Excel. Установить дату начала проекта 19 апреля 2007 года. Определить рабочее время проекта (без изменений). Перейти к ссылке «Планирование задач» и ввести связи типа «окончание–начало» (кнопкой на панели инструментов) в соответствии с заданным сетевым графиком.



Проверить список предшественников.

Корректировка списка задач и формирование структуры проекта

Цель работы: умение вносить изменения в список задач проекта, умение формировать структуру проекта путем повышения и понижения уровня задач проекта.

Корректировка списка задач проекта

Во время работы над проектом довольно часто возникает необходимость внести изменения в список задач проекта: добавить новую задачу (в том числе не только в конец списка), удалить ошибочно введенную, изменить порядок следования задач.

Добавление задачи

Чтобы добавить в список новую задачу, надо выделить задачу (щелкнуть левой кнопкой мышки в поле задачи), перед которой нужно поместить новую задачу, и из меню **Правка** выбрать команду **Вставить**. В результате этих действий в список задач будет добавлена пустая строка, в которую можно ввести новую задачу.

Удаление задачи

Чтобы удалить задачу, надо выделить эту задачу (щелкнуть левой кнопкой мышки в поле задачи) и из меню **Правка** выбрать команду **Удалить**.

Перемещение задачи

Чтобы переместить задачу (или группу следующих одна за другой задач) в другое место списка, надо выделить нужную задачу (задачи), и из меню **Правка** выбрать команду ***Вырезать***. Затем выделить задачу, перед которой надо поместить выделенные на предыдущем шаге задачи, и из меню **Правка** выбрать команду ***Вставить***.

Формирование структуры проекта

Представление задач в проекте в виде простого списка недостаточно наглядно. Простой список не отражает структуру проекта, связи между задачами, не позволяет видеть главные и подчиненные задачи. Гораздо удобнее задачи проекта представить в виде иерархического списка, в котором задачи разделены по уровням.

Обычно во время работы над проектом сначала формулируется цель проекта (главная задача), затем она разбивается на фазы (крупные задачи), фазы разбиваются на задачи, задачи — на подзадачи более низкого уровня и т. д. до тех пор, пока не будут определены

все необходимые для завершения проекта задачи. Таким образом, проект можно рассматривать как совокупность обобщенных и подчиненных задач. Обобщенная задача – своего рода заголовок, она суммирует стоимость и длительность задач нижнего уровня. Подчиненная задача — это часть обобщенной задачи.

Перевести задачу с одного уровня на другой, сделать ее подчиненной или главной (если задача уже является подчиненной) можно в режиме просмотра диаграммы Ганта, для перехода в который надо из меню **Вид** выбрать команду **Диаграмма Ганта** или в режиме просмотра списка задач. В этих режимах на панели инструментов появляются кнопки, позволяющие формировать структуру проекта.

В режиме просмотра диаграммы Ганта, рядом со списком задач выводится диаграмма Ганта, на которой подчиненные задачи изображаются горизонтальными столбиками, обобщенные — скобками. При этом, если подчиненные задачи начинаются одновременно, то длительность обобщенной задачи полагается равной длительности наиболее длительной подчиненной задачи.

Чтобы понизить уровень задачи, сделать ее подчиненной, надо выделить эту задачу и щелкнуть на кнопке со стрелкой вправо.

Чтобы повысить уровень задачи, сделать ее обобщенной, для задач за ней следующих, надо выделить эту задачу и щелкнуть на кнопке со стрелкой влево.

Следует обратить внимание, что при копировании или перемещении обобщенной задачи все подчиненные задачи также копируются или перемещаются. Если надо переместить только обобщенную задачу, то сначала надо перевести подчиненные задачи на уровень обобщенной. Проект можно просматривать с различной степенью детализации.

Щелкните на кнопке со значком минус, чтобы скрыть подчиненные задачи текущей выделенной, задачи. Если задача является обобщенной и подчиненные задачи скрыты, то нажмите на кнопку со значком плюс, чтобы просмотреть подчиненные задачи. Чтобы увидеть все скрытые задачи проекта, нажмите на кнопку с двумя плюсами.

***Задание 2*:**

1. Откройте файл «Учебный проект».
2. Добавьте в начало списка задач название вашего проекта и сделайте все введенные ранее задачи подчиненными этой задаче.
3. Сформируйте структуру своего проекта: определите главные и подчиненные задачи.
4. Просмотрите задачи проекта. Внесите в него необходимые изменения (например, добавьте несколько новых, уточняющих задач).
5. Научитесь просматривать проект с различной степенью детализации.
6. Сохраните измененный проект.

Назначение связей между задачами

Цель работы: знание типов связей между задачами, понятий «время опережения» и «время задержки»; умение связывать задачи проекта связями различного типа.

Связи между задачами

Задачи реального проекта связаны между собой во времени. Например, некоторые задачи могут выполняться одновременно, другие не могут быть начаты до тех пор, пока не завершится некоторая предыдущая задача. Поэтому после того, как составлен список задач и задачи распределены по уровням (определены обобщенные задачи и подзадачи), необходимо установить связи между задачами.

Различают три типа связей между задачами проекта:

- конец – начало (finish-to-start, FS);
- начало – начало (start-to-start, SS);
- конец – конец (finish-to-finish, FF).

Чтобы назначить связь типа «конец–начало» между следующими в списке друг за другом задачами, надо выделить эти задачи и щелкнуть на командной кнопке **Связать задачи.**

Время задержки (опережения) выполнения задачи

Иногда между завершением одной задачи и началом другой должно пройти некоторое время, или задача-приемник должна начинаться немного раньше, чем завершится задача-родитель. Например, пол, после покрытия лаком должен сохнуть 48 часов. Следовательно,

задача «оборудование» может начаться только через 48 часов после завершения задачи «окраска».

Другой пример. Если монтируется линия электропередачи из 30 столбов, то совсем не обязательно ждать, пока будут установлены все столбы, чтобы приступить к монтажу проводов. Задачу монтажа можно начать после того, как будут установлены, например, 5 столбов. Подобные ситуации моделируются заданием времени отставания для задачи-приемника. Для задания времени отставания (опережения) используют форму. Время отставания задается вводом положительного числа в поле формы, опережения – отрицательного.

Временной масштаб диаграммы Ганта

Диаграмму Ганта удобно использовать для наблюдения связей между задачами проекта. В левой части окна отображается список задач проекта, в правой — диаграмма Ганта. При этом критические задачи, длительность которых оказывает влияние на длительность всего проекта, отображаются красным цветом.

Если проект длительный, то диаграмма Ганта не помещается в одном окне. Это не всегда удобно. Microsoft Project позволяет настроить временной масштаб диаграммы таким образом, чтобы в окне были видны сразу все задачи проекта.

В верхней части диаграммы Ганта выводится временная ось. Масштаб оси устанавливается в диалоговом окне, которое появляется при выборе из меню **Формат**.

Контрольные вопросы

1. Перечислите типы связей, которыми могут быть связаны задачи проекта.
2. Если некоторая задача может быть начата только через некоторое время после завершения другой (например, из технологических соображений), то как этот факт отображается в модели проекта?
3. Если некоторая задача должна начинаться несколько позже, чем закончится другая задача проекта, то как этот факт отразить в модели проекта?
4. Если длительность проекта такова, что диаграмма Ганта не помещается в пределах экрана, то что и как надо сделать, чтобы вся диаграмма Ганта была видна одновременно?

***Задание 3*:**

1. Откройте файл «Учебный проект».
2. Установите связи между задачами проекта таким образом, чтобы модель проекта соответствовала реальному проекту.
3. Найдите в проекте задачи, которые должны выполняться с задержкой относительно других задач, и задайте для них время задержки.
4. Найдите в проекте задачи, которые могут выполняться одновременно с другими задачами, но с некоторым опережением начала их выполнения. (Задайте для этих задач время опережения.)
5. Установите такой временной масштаб диаграммы Ганта, чтобы вся диаграмма была видна в одном экране.
6. Сохраните файл проекта.

Ресурсы проекта

Цель работы: знание понятия «ресурс», характеристик ресурса как составной части проекта, умение составлять список ресурсов проекта.

Цена ресурса и стоимость задачи

Для выполнения задач проекта необходимы ресурсы: люди, оборудование, механизмы. За использование ресурса для реализации задачи надо платить. Стоимость использования ресурса определяется ценой ресурса и временем его использования. Цена ресурса измеряется в денежных единицах, отнесенных к единице времени (день, час, минута). Например, цена ресурса «грузовик» может быть 500 руб./ч. Это означает, что если для выполнения некоторой задачи необходимо использовать ресурс «грузовик» в течение двух часов, то стоимость этого ресурса равна 1000 руб.

Суммарная стоимость ресурсов, использованных для реализации задачи, определяет стоимость этой задачи. Из стоимости ресурсов, использованных для реализации задач проекта, складывается стоимость проекта в целом.

Ресурсы проекта

Работая над проектом, нужно иметь список доступных ресурсов. Наиболее легко составить и просмотреть список ресурсов проекта можно в режиме «Список ресурсов».

Контрольные вопросы

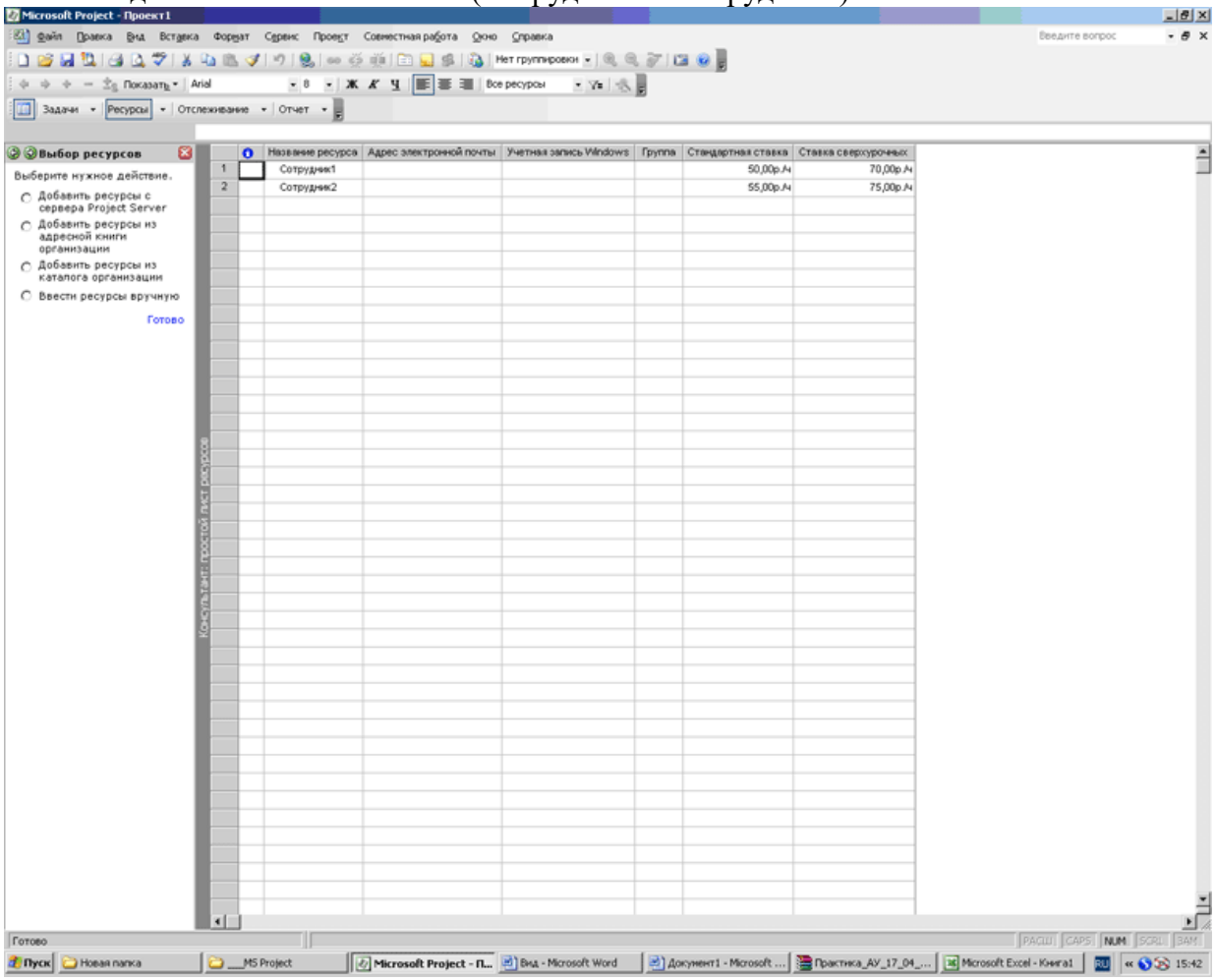
1. Перечислите характеристики ресурса.
2. Чем определяется стоимость задачи проекта?
3. Чем определяется стоимость проекта в целом?

Задание 4:

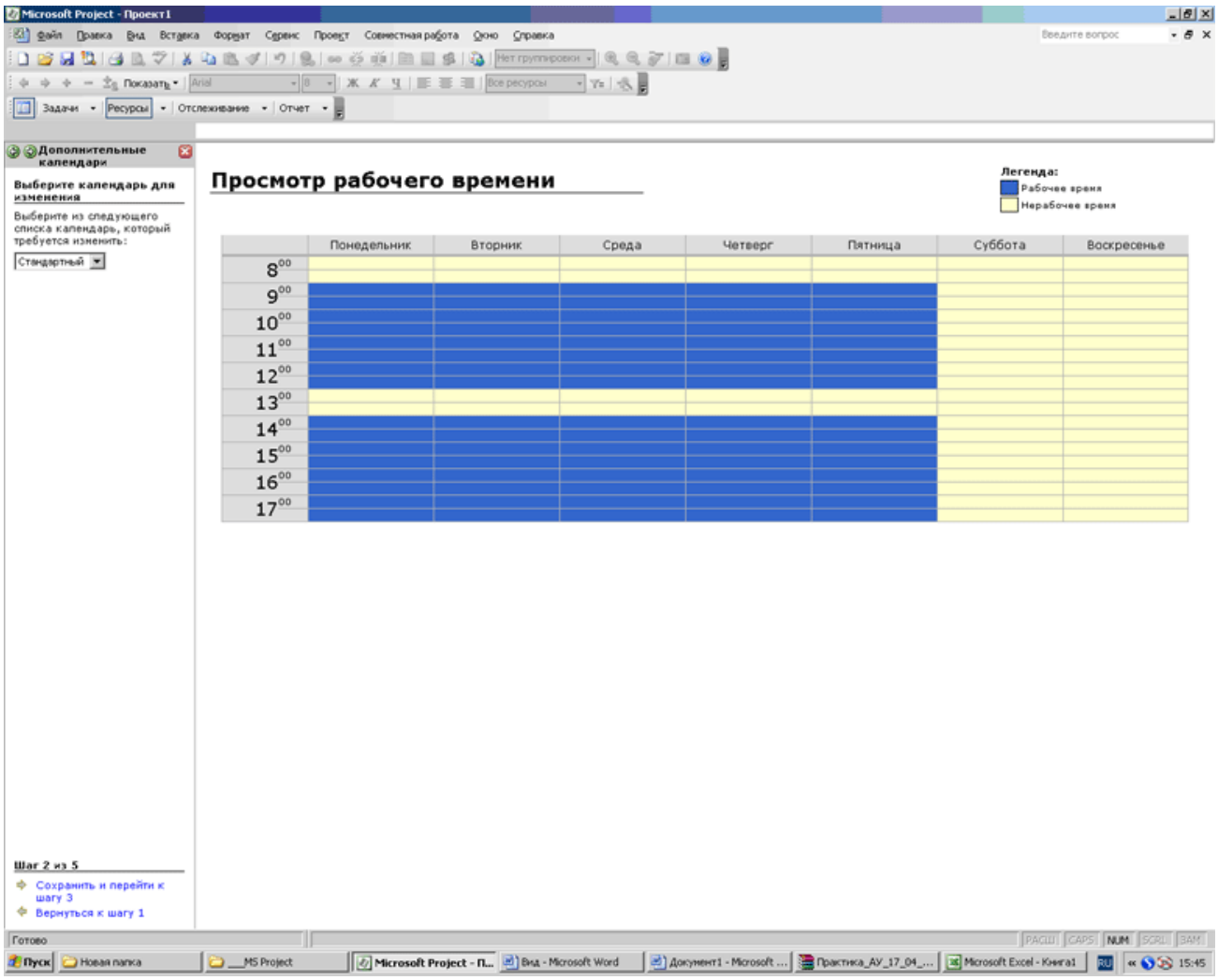
1. Откройте файл «Учебный проект».
2. Составьте список ресурсов, необходимых для реализации вашего проекта. Для каждого ресурса задайте цену, цену при сверхурочном использовании, величину фиксированной платы за ресурс, количество ресурса, доступное в рамках реализации проекта и другие характеристики.
3. Сохраните измененный проект.

Пример выполнения

Ввести данные об исполнителях (Сотрудник1 и Сотрудник2).



Каждому сотруднику определить рабочее время (пока без изменений).



Назначение ресурсов задачам проекта

Цель работы: умение назначать ресурсы задачам проекта.

Стоимость и длительность реализации задачи

После того, как сформирована структура проекта (составлен список задач, определены главные и подчиненные задачи, назначены связи между задачами) и составлен список ресурсов проекта, можно приступить к назначению ресурсов задачам проекта. В результате назначения ресурсов задачам будет определена стоимость и длительность задач проекта.

Стоимость задачи определяется стоимостью ресурсов, необходимых для ее реализации. Чем больше ресурсов требует задача, тем она дороже, тем больший вклад вносит она в стоимость реализации проекта в целом.

Длительность реализации задачи зависит от количества единиц ресурса, выделенных для ее реализации. Здесь надо вспомнить, что на этапе составления списка задач длительность каждой задачи задавалась в предположении, что она реализуется одной единицей ресурса. После назначения ресурсов длительность выполнения задачи определяется уже количеством единиц ресурсов, выделенных для ее решения.

Назначение ресурсов

Время реализации проекта зависит от длительности реализации задач, которая в свою очередь зависит от количества ресурсов, выделенных для их реализации. Таким образом, чем больше ресурса будет выделено задаче, тем быстрее она будет выполнена. Следует обратить внимание, что некоторые задачи, которые могут выполняться одновременно, могут требовать одинаковых ресурсов. При этом возникает проблема распределения ресурсов между задачами: как распределить ресурсы таким образом, чтобы время реализации всего проекта было минимальным? Очевидно, что в первую очередь ресурсы надо выделять задачам, длительность которых определяет длительность реализации всего проекта (такие задачи называются критическими и на диаграмме Ганта изображаются красными прямоугольниками).

Чтобы назначить ресурс задаче, надо выделить, например в режиме просмотра диаграммы Ганта, задачу, которой назначается ресурс, и щелкнуть на кнопке «Ресурс».

Следует обратить внимание, что при назначении задачи одной единицы ресурса, значения количества работы и длительности задачи совпадают. Если задаче назначить несколько единиц ресурса, то длительность задачи сокращается пропорционально количеству назначенных единиц, при этом величина количества работы не изменяется.

Одной задаче можно назначить несколько разных ресурсов. Однако следует обратить внимание, что на длительность задачи влияет только первый из назначенных ресурсов. Остальные только изменяют стоимость задачи.

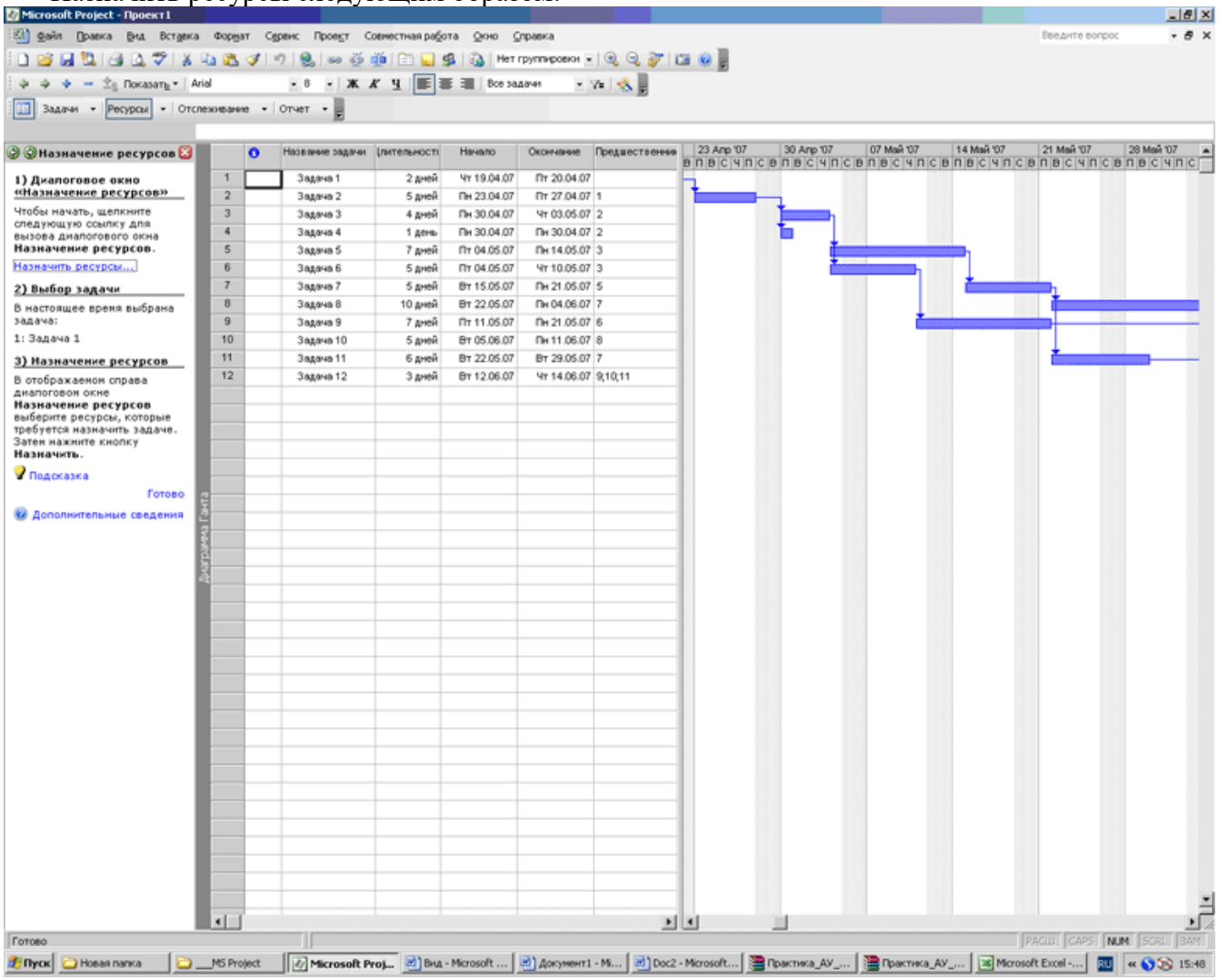
После назначения задачам ресурсов, необходимо убедиться, что ресурсы назначены правильно, например, что некоторый ресурс не назначен нескольким задачам, которые выполняются одновременно, или какой-либо задаче не назначено больше единиц ресурса, чем доступно для реализации проекта.

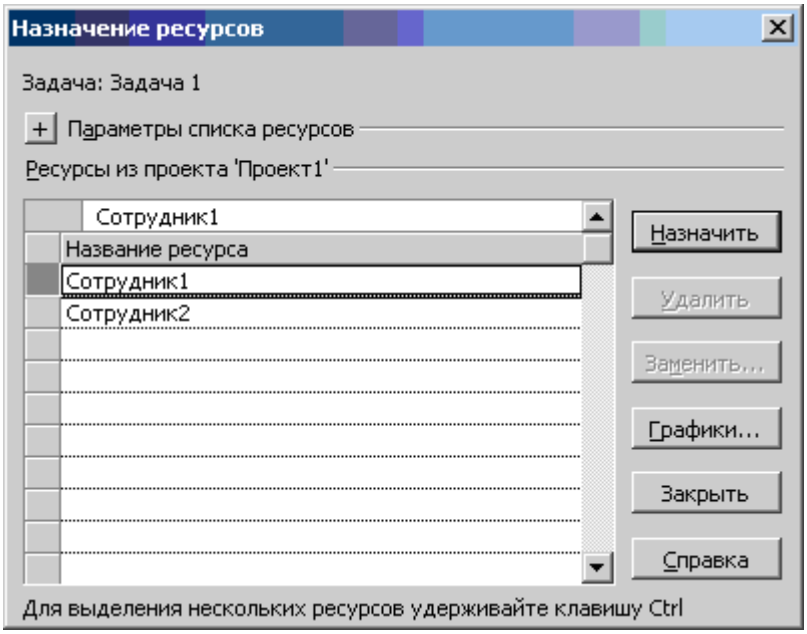
***Задание 4*:**

1. Откройте файл проекта, над которым вы работаете.
2. Назначьте задачам проекта ресурсы. Используя режим просмотра списка ресурсов, убедитесь, что ресурсы задачам назначены верно.
3. Сохраните измененный проект.

Пример выполнения

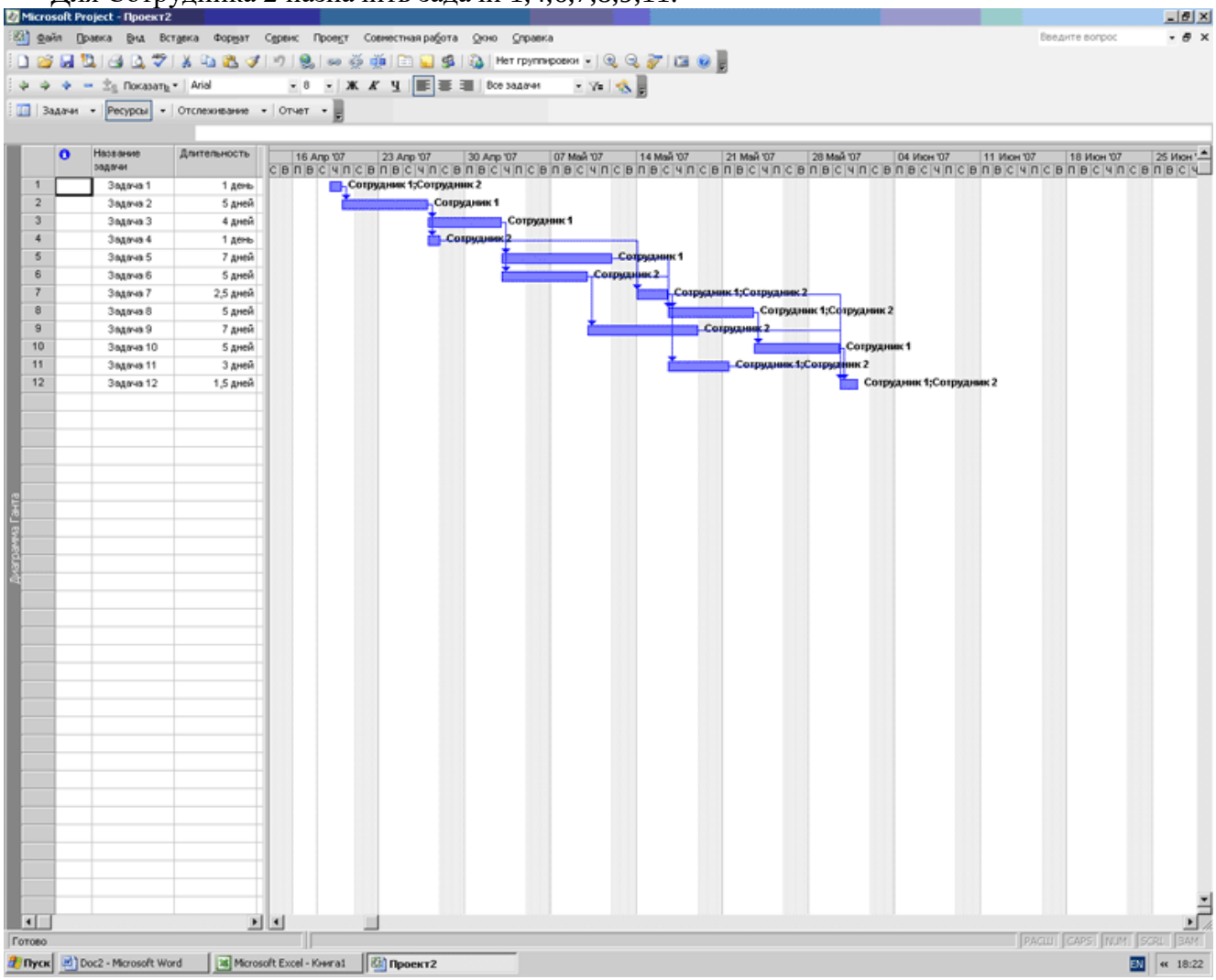
Назначить ресурсы следующим образом:





Для Сотрудника 1 выделить задачи 1,2,3,5,7,8,10,11 (удерживая клавишу Ctrl) и нажать кнопку «Назначить».

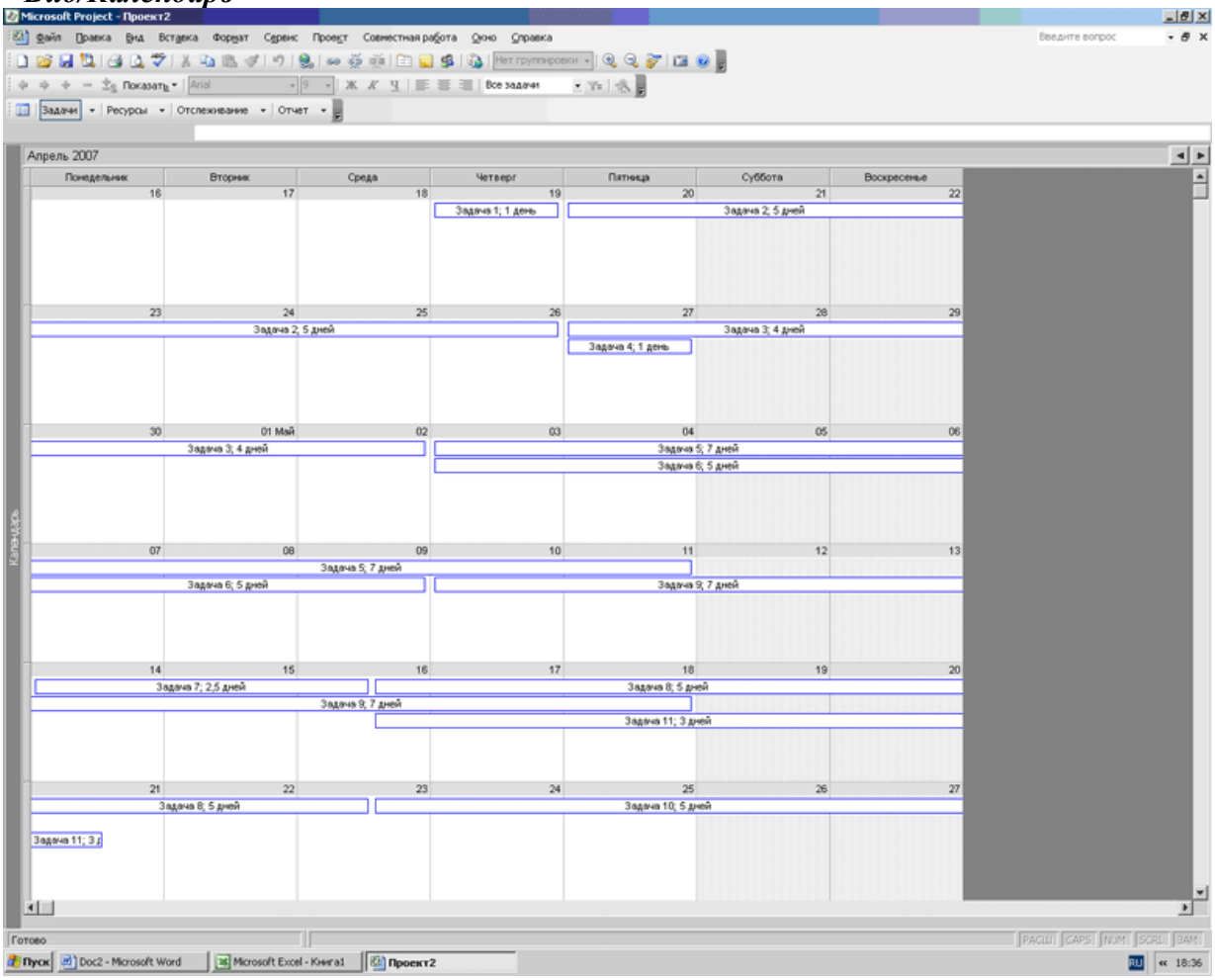
Для Сотрудника 2 назначить задачи 1,4,6,7,8,9,11.



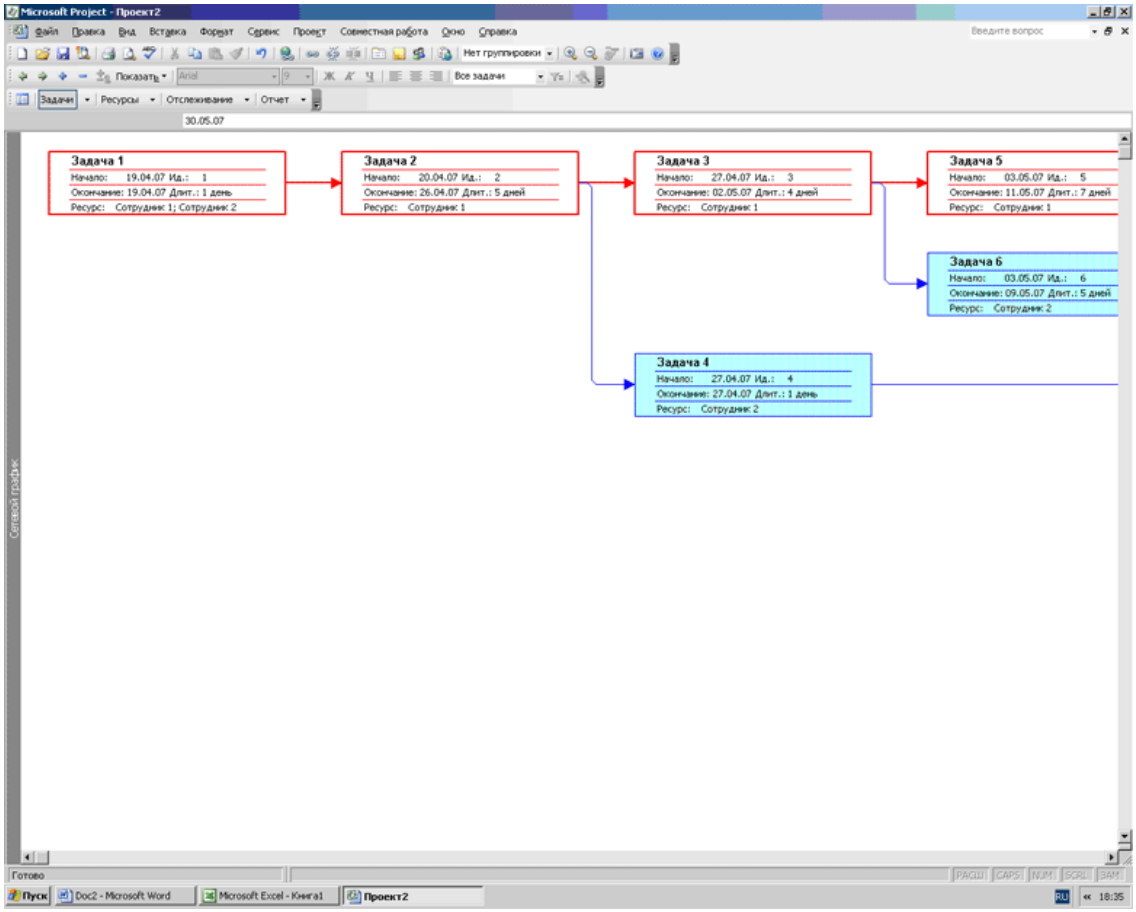
Варианты представления проекта (меню «Вид»)

Графические варианты представления проекта очень разнообразны. Примеры представлены далее.

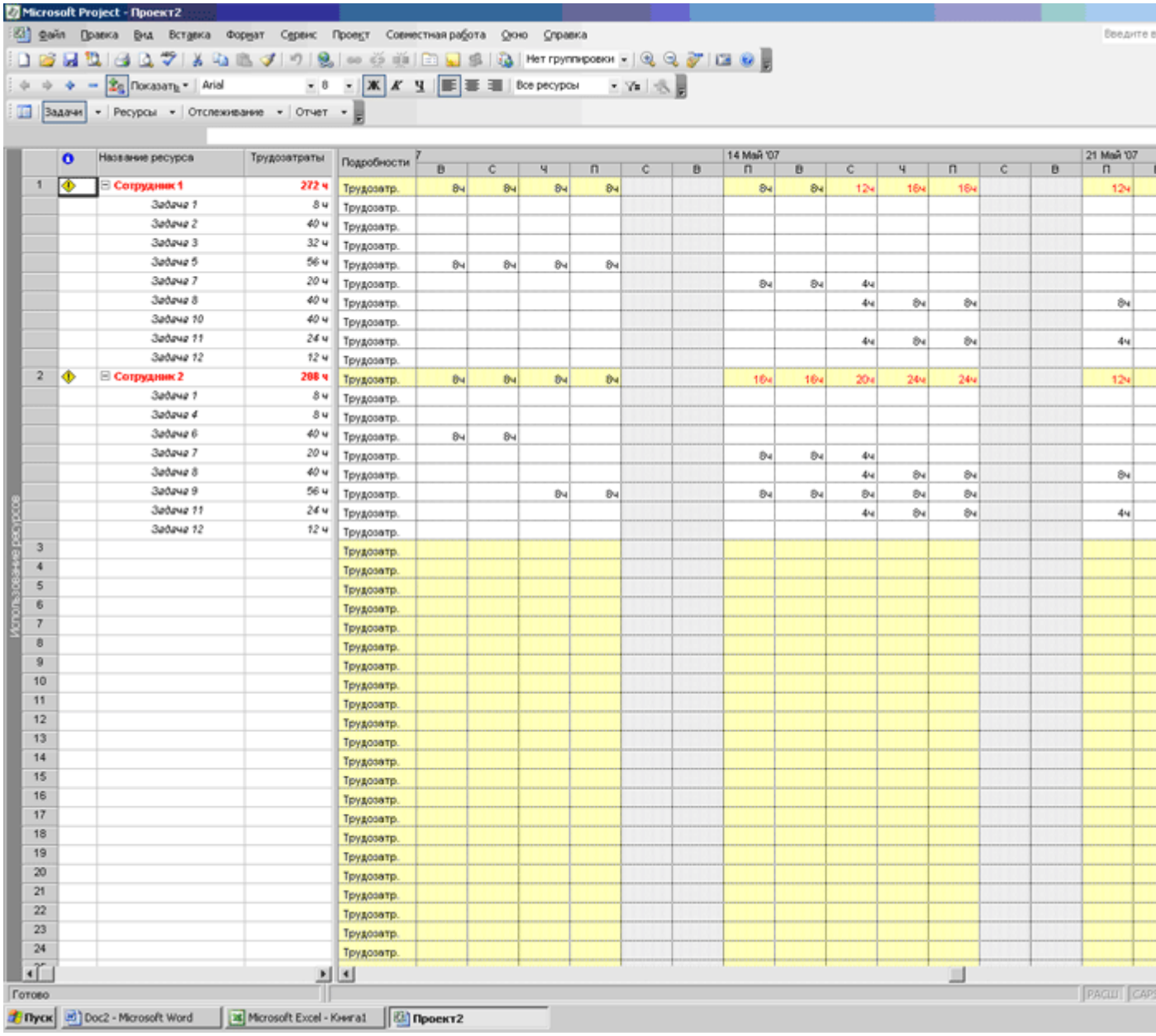
Вид/Календарь



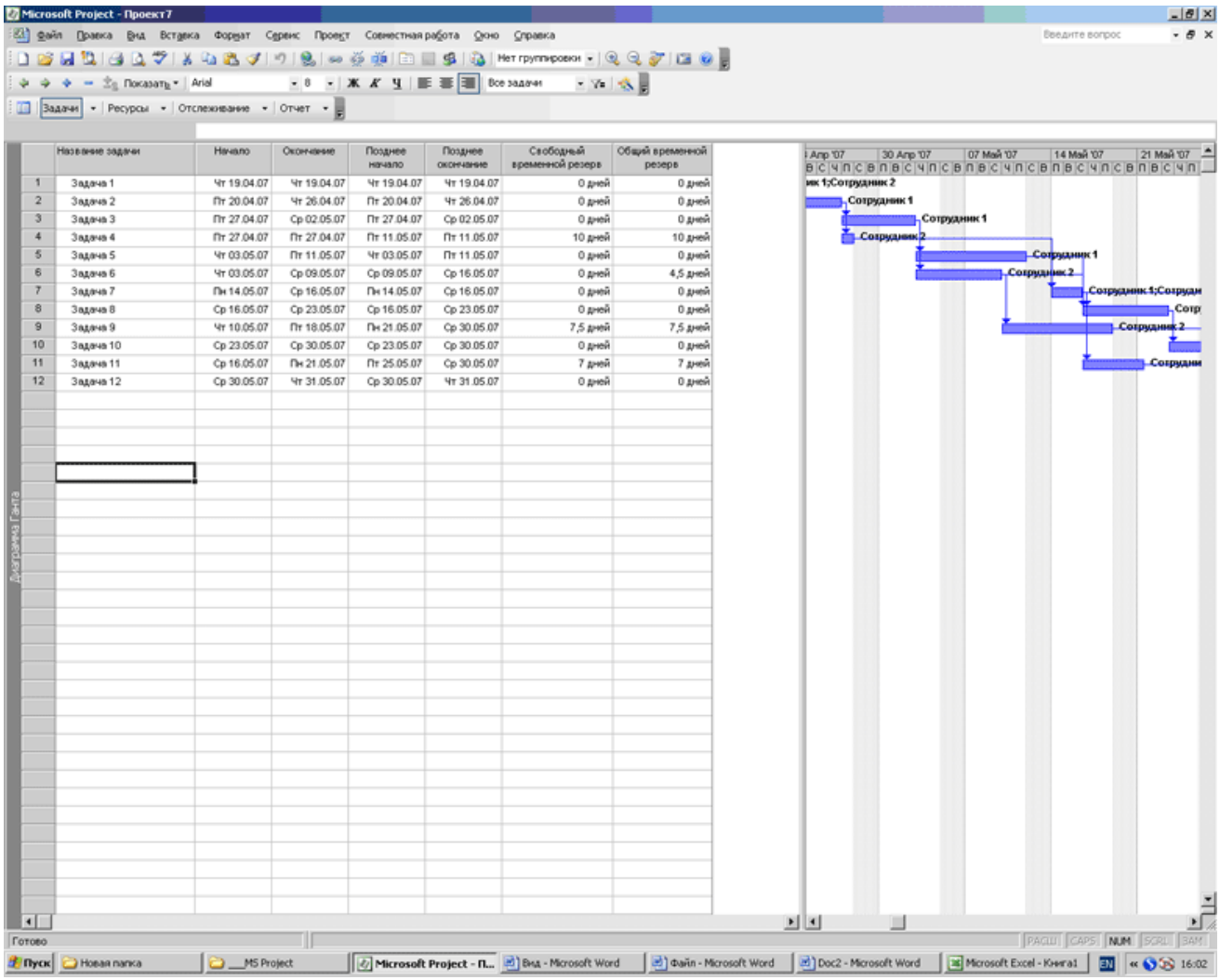
Вид/Сетевой график



Вид/Использование ресурсов

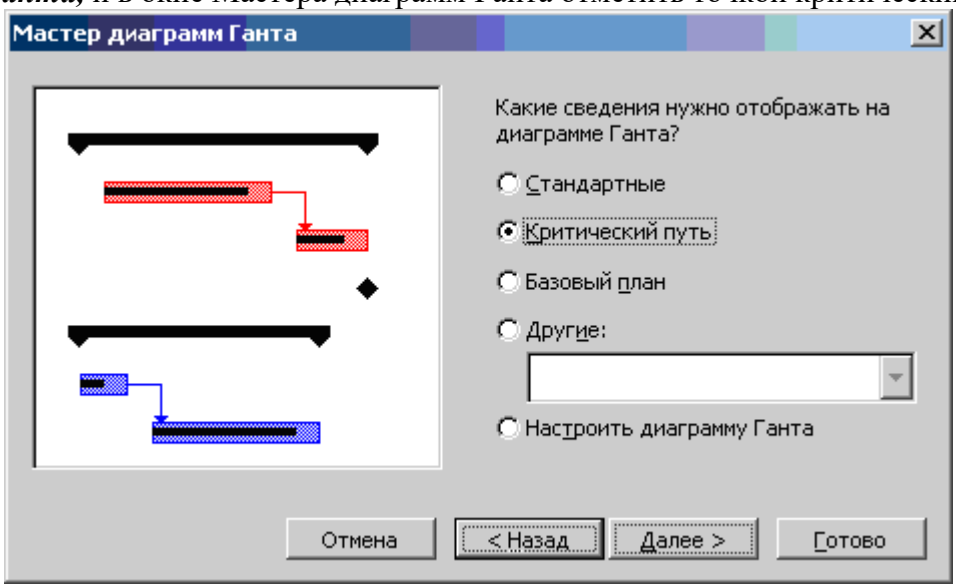


Вид/Таблица.../Календарный план

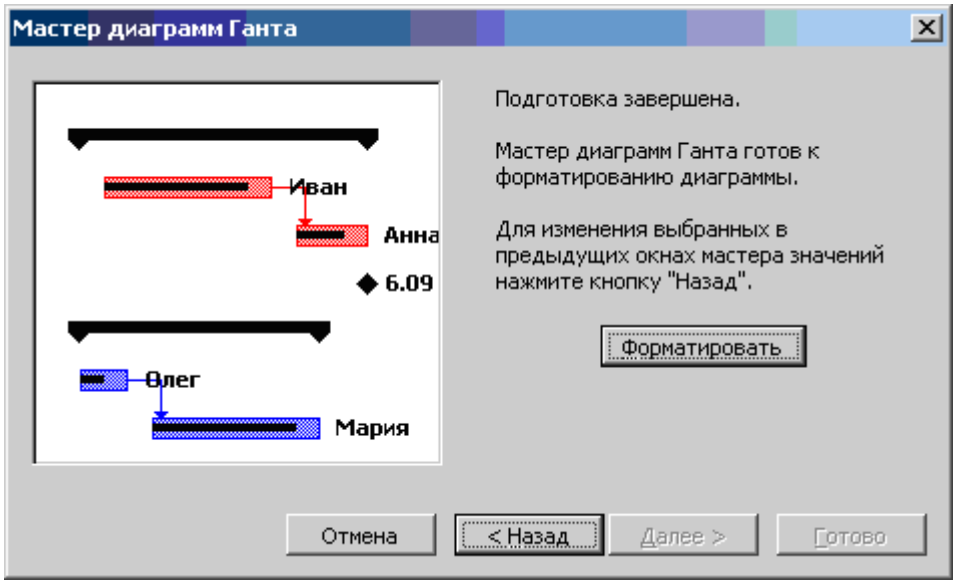


Просмотр критического пути

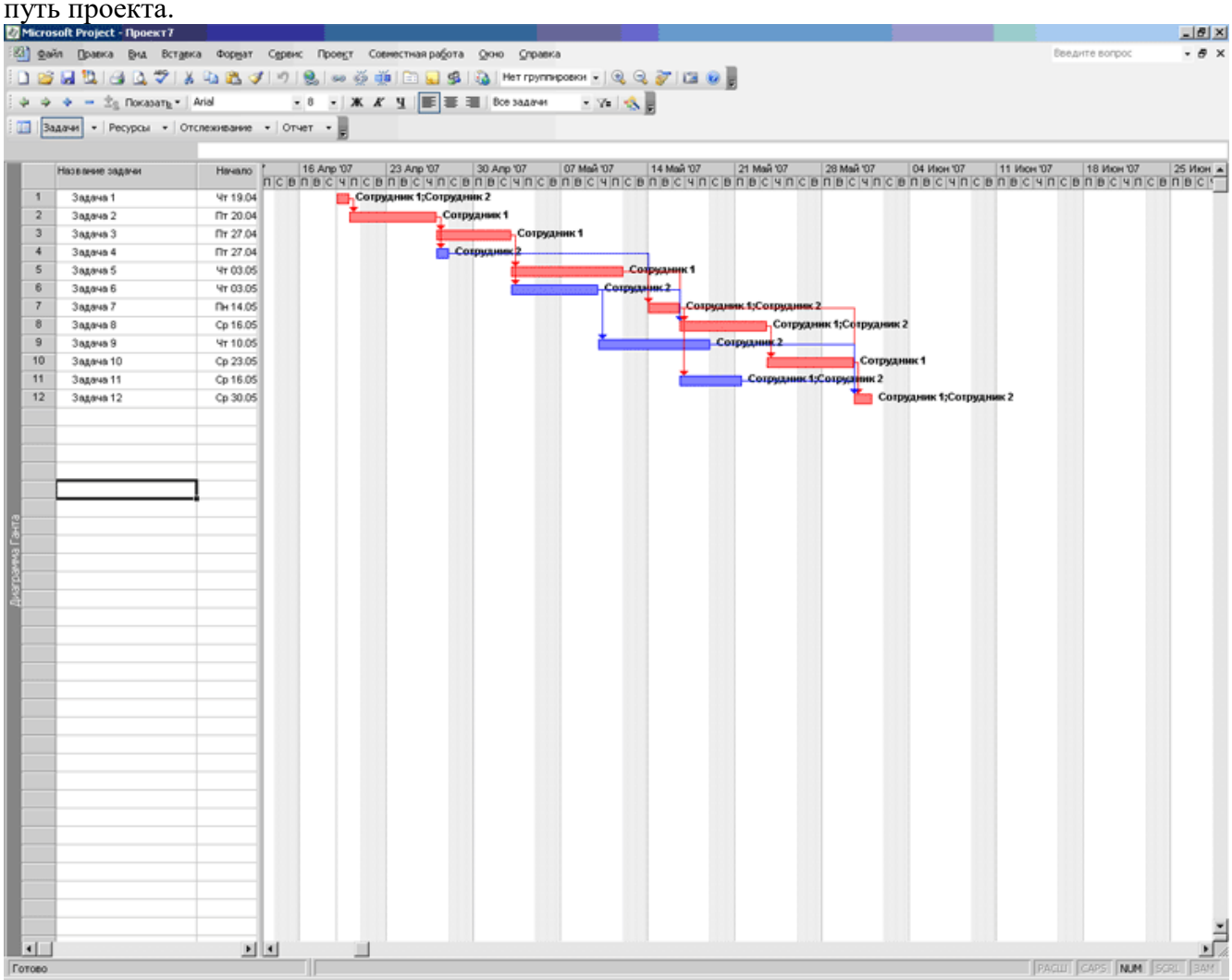
Для вывода на экран критического пути необходимо открыть меню ***Формат/Мастер диаграмм Ганта,*** и в окне Мастера диаграмм Ганта отметить точкой критический путь.



Выбрать «Далее».

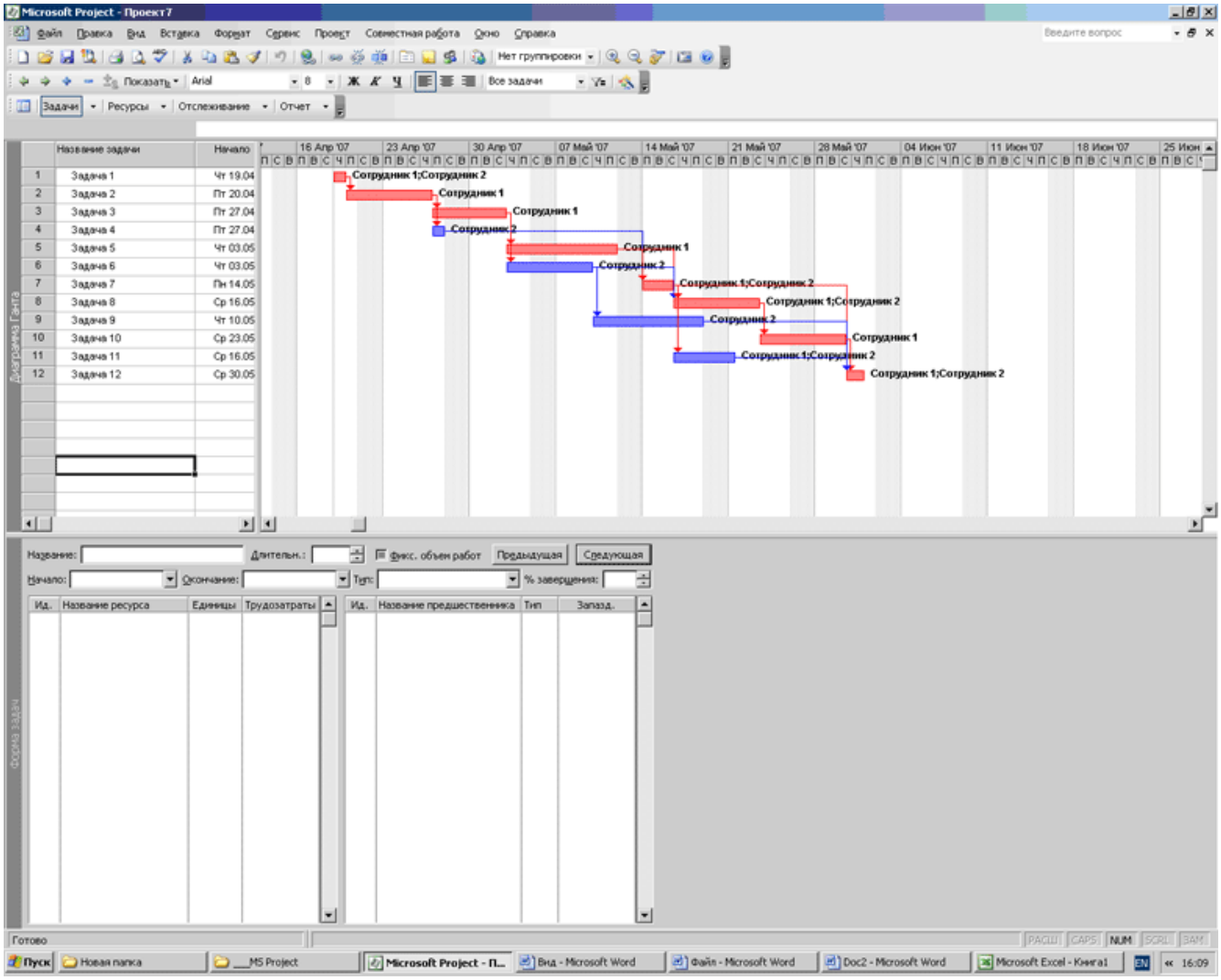


Выбрать «Форматировать» и «Выход». В результате в окне будет отражен критический путь проекта.

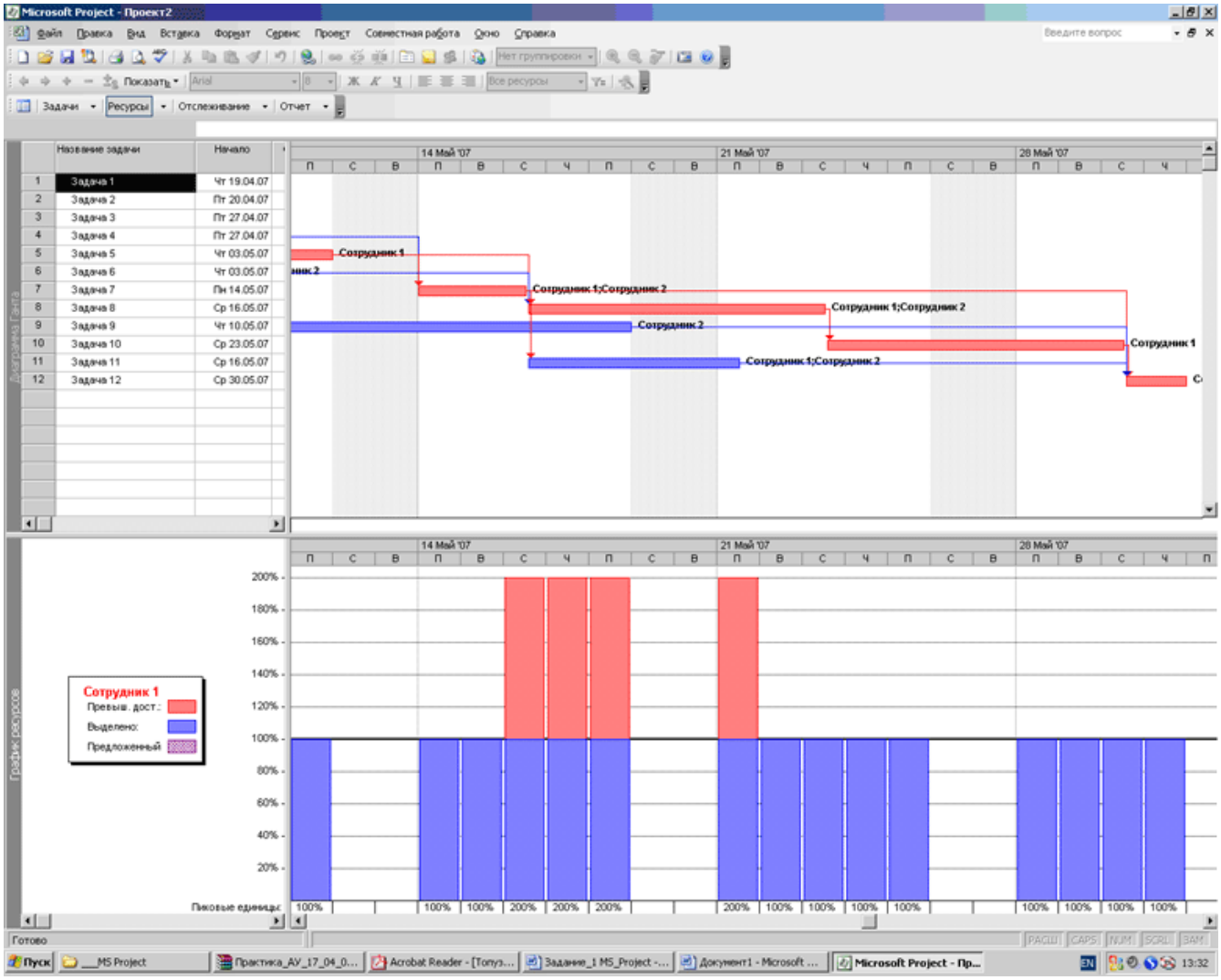


Просмотр графика загрузки ресурсов

Для просмотра графика загрузки ресурсов выбрать меню ***Окно/Разделить.***



Щелкнуть по надписи ***Форма задач.*** Выбрать меню ***Вид/График ресурсов***.



Результатом будет график загрузки ресурсов.

Устранение перегрузки ресурсов

В процессе выполнения предыдущего задания было выяснено, что и сотрудник 1, и сотрудник 2 работают с перегрузкой на временном отрезке 14 мая — 21 мая.

Далее начинается работа по устранению перегрузок ресурсов. Для этого в MS Project есть несколько путей, в частности:

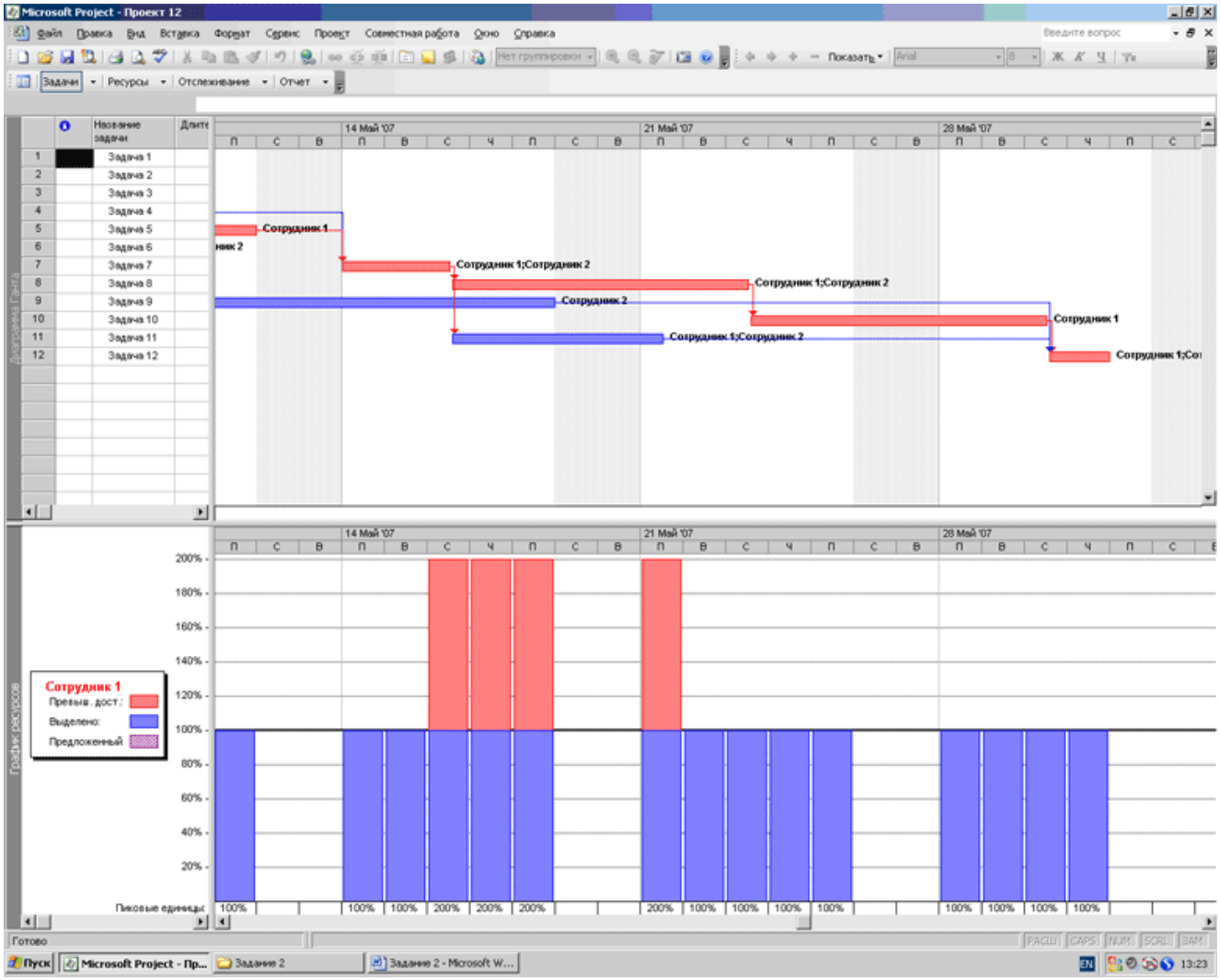
- перенос (сдвиг) времени выполнения отдельных задач (только не лежащих на критическом пути);
- прерывание выполнения отдельных задач;
- добавление сверхурочных работ (что не устраняет перегрузки, но по-другому оплачивается);
- назначение работы в выходные дни;
- добавление в список ресурсов новых сотрудников и др.

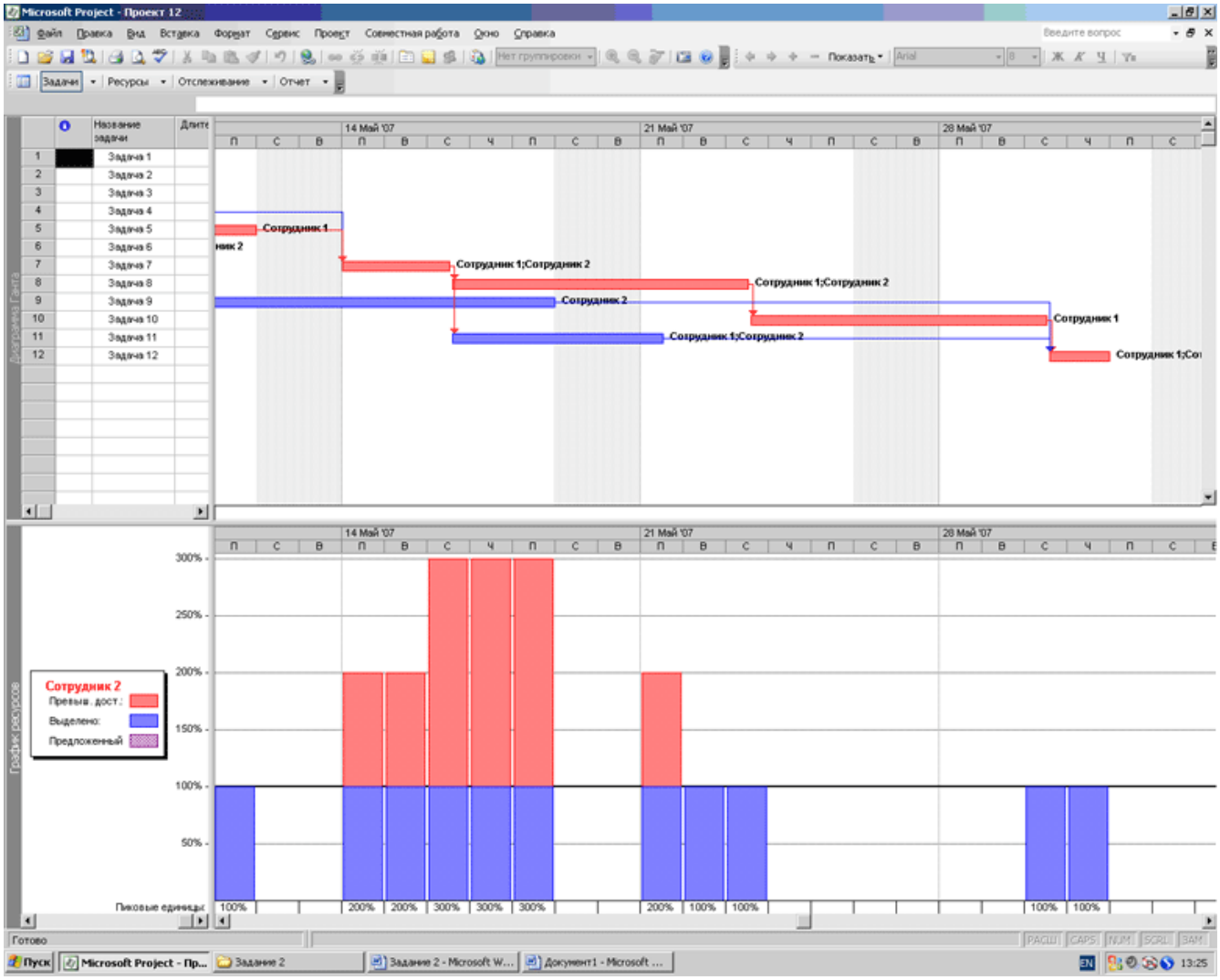
Все эти методы требуют предварительного анализа ситуации. И не всегда приводят к отличному результату.

Поэтому необходимо оставить один файл в качестве образца, а все изменения проекта производить на его копиях.

Рассмотрим конкретные примеры.

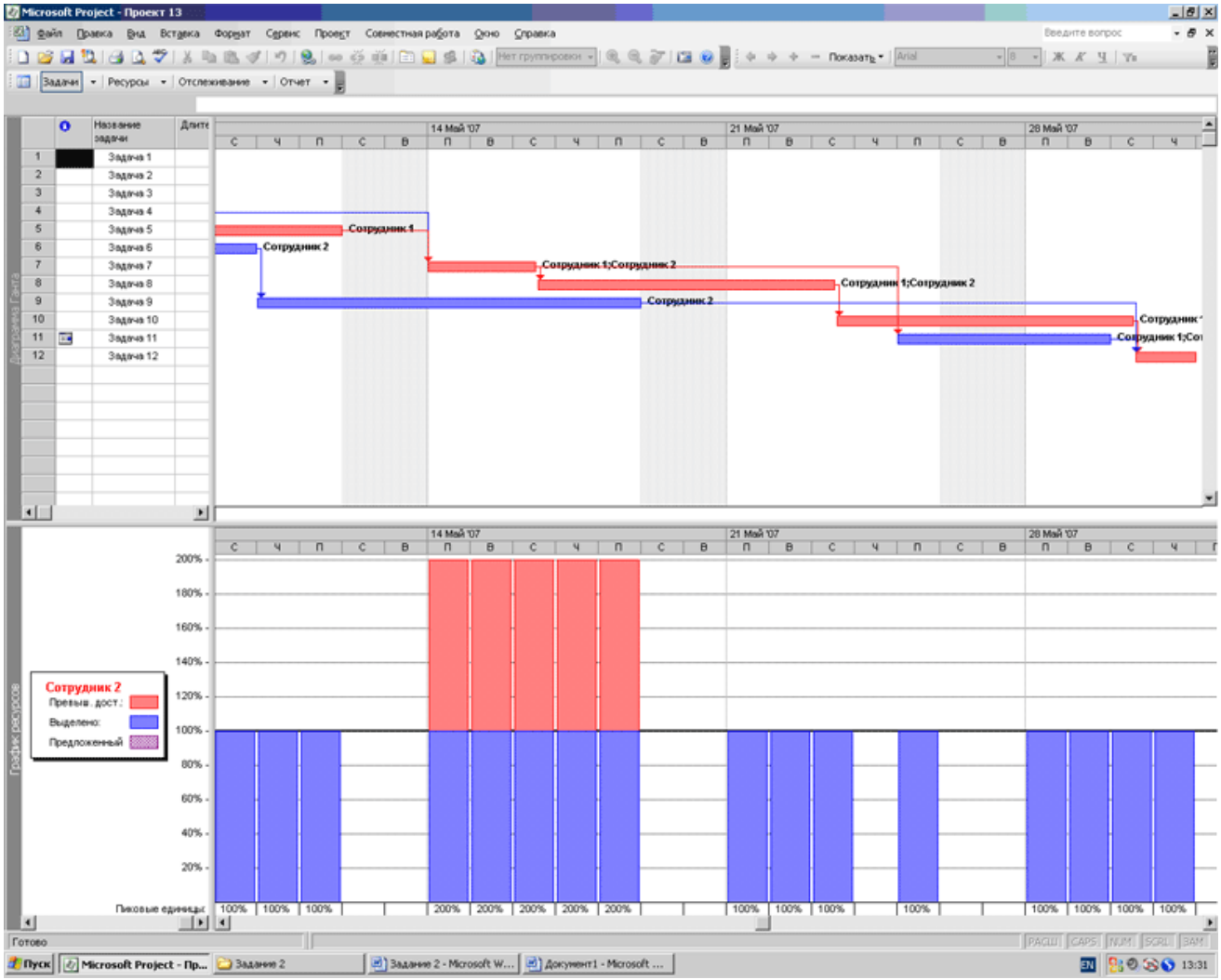
Перенос отдельной задачи на более поздний срок


Microsoft Project - Проект 12
Файл Правка Вид Вставка Формат Сервис Проект Совместная работа Окно Справка
Задачи Ресурсы Отслеживание Отчет
Название задачи
Задача 1
Задача 2
Задача 3
Задача 4
Задача 5
Задача 6
Задача 7
Задача 8
Задача 9
Задача 10
Задача 11
Задача 12
14 Май '07
21 Май '07
28 Май '07
Сотрудник 1
Сотрудник 1;Сотрудник 2
Сотрудник 2
Диаграмма Ганта
График ресурсов
Сотрудник 1
Превыш. дост.:
Выделено:
Предложенный
200%
180%
160%
140%
120%
100%
80%
60%
40%
20%
Пиковые единицы:
Готово
Пуск
Microsoft Project - Пр...
Задание 2
Задание 2 - Microsoft W...
13:23

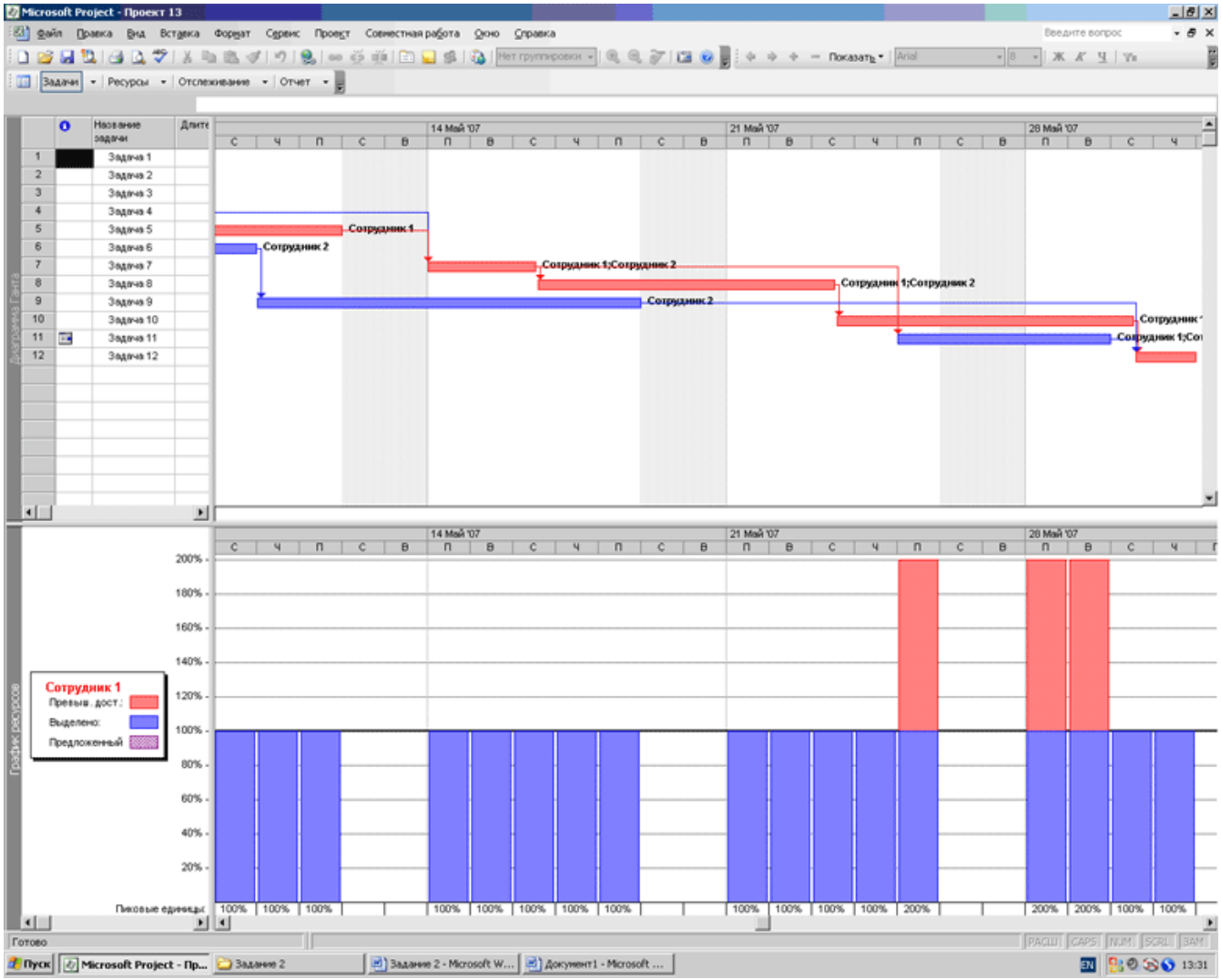


Из анализа графика сотрудника 1 ясно, что выполняемые им задачи передвигать не имеет смысла, так как этот сотрудник всегда занят на 100 %.

Для сотрудника 2 можно попробовать перенести задачу 11 на более поздний срок.

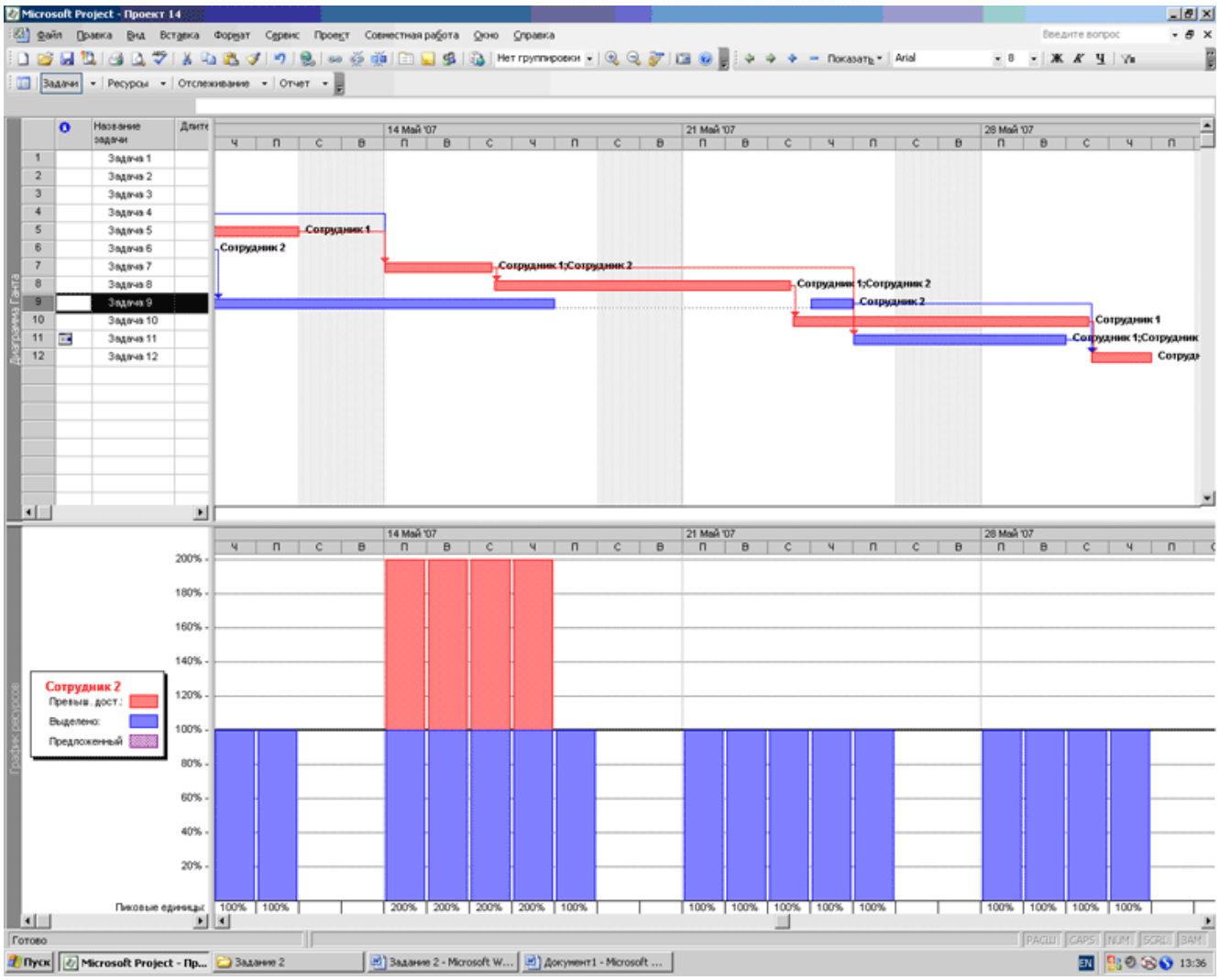


Часть перегрузки устранена. Осталось 5 дней с перегрузкой не более 200 % (ранее было и 300 %). Кроме того, график сотрудника 1 тоже изменился в лучшую сторону.



Прерывание выполнения задачи

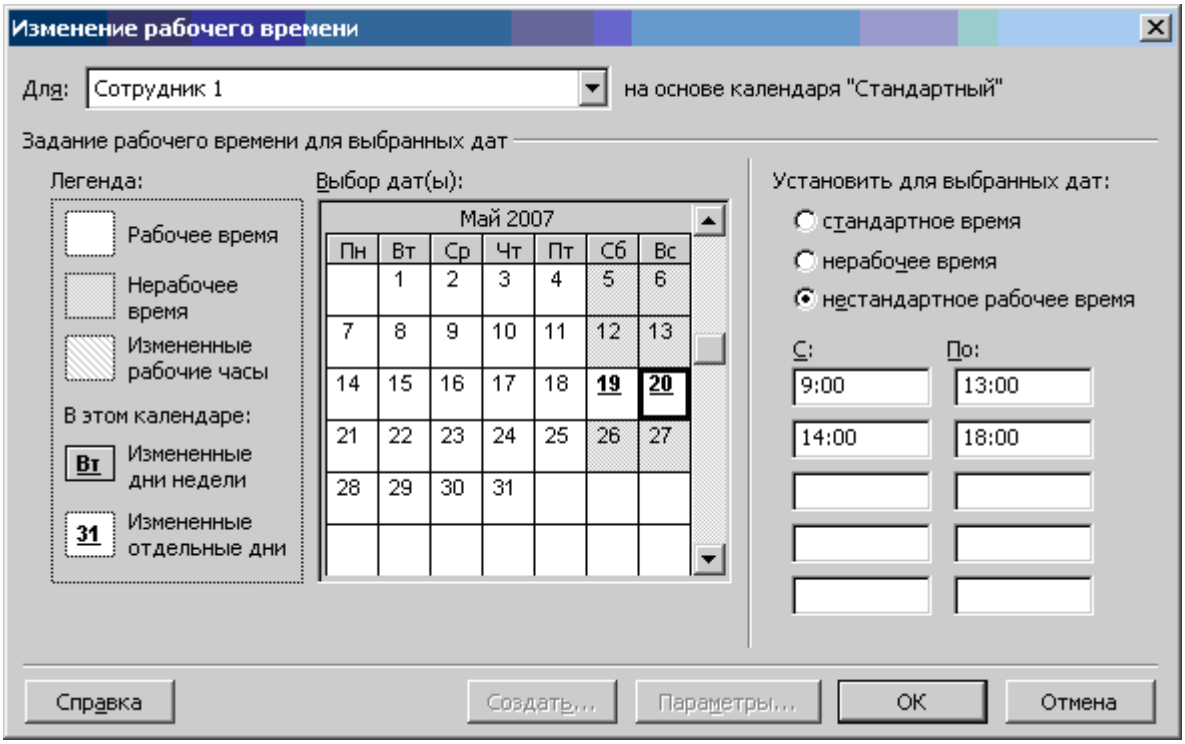
По графику сотрудника 2 видно, что 24 мая он не выполняет никаких работ. Поэтому задачу 9 можно прервать (есть кнопка на панели инструментов) и перенести на этот день.



Таким образом, в результате переноса и прерывания двух задач перегрузка сотрудников уменьшена примерно в 2 раза.

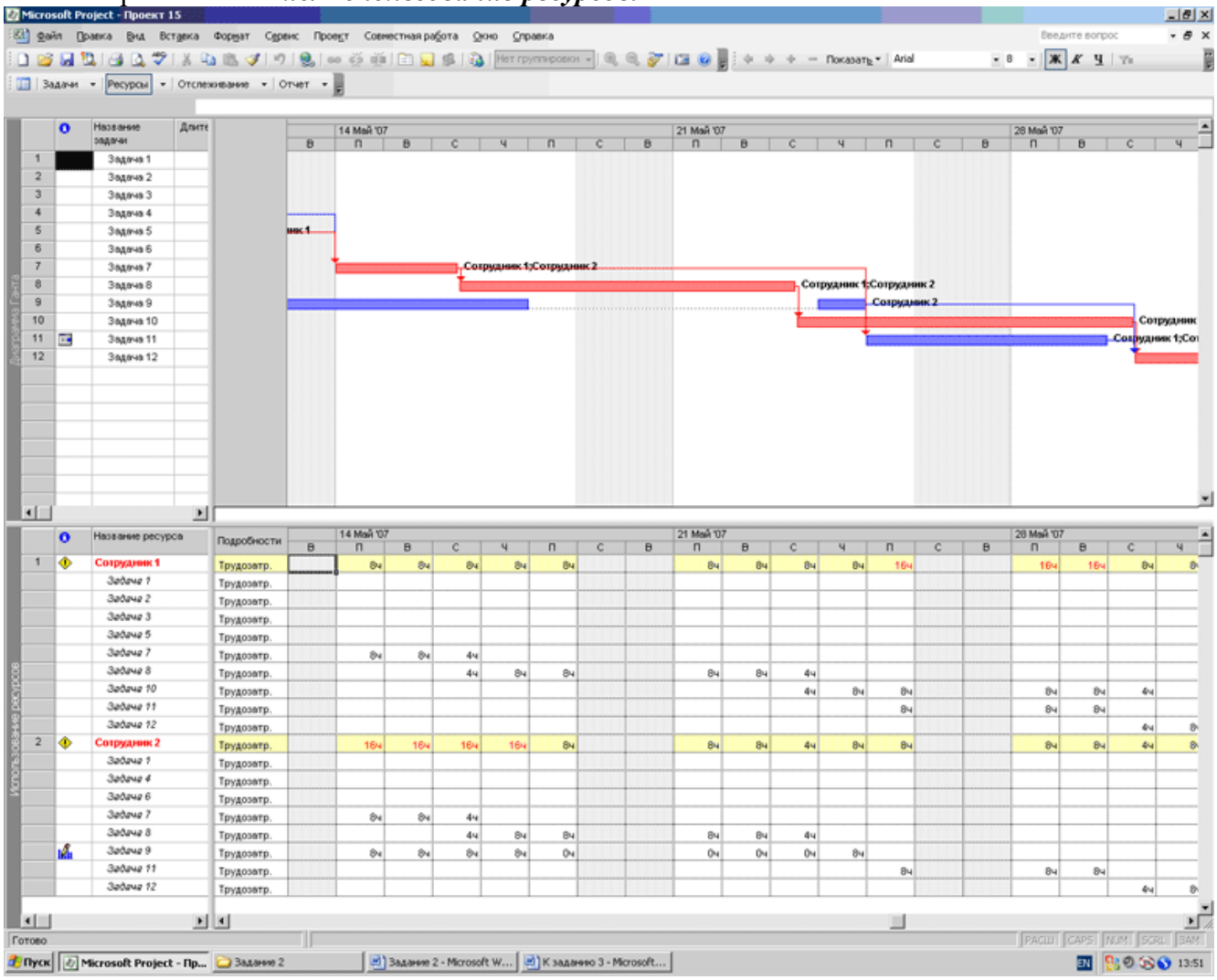
Использование выходных дней

Для этого варианта необходимо для конкретного сотрудника перейти в режим «Изменение рабочего времени» и установить там нестандартное рабочее время в выходные. ***Определите, какие выходные наилучшим образом для этого подходят с точки зрения уменьшения перегрузки.***

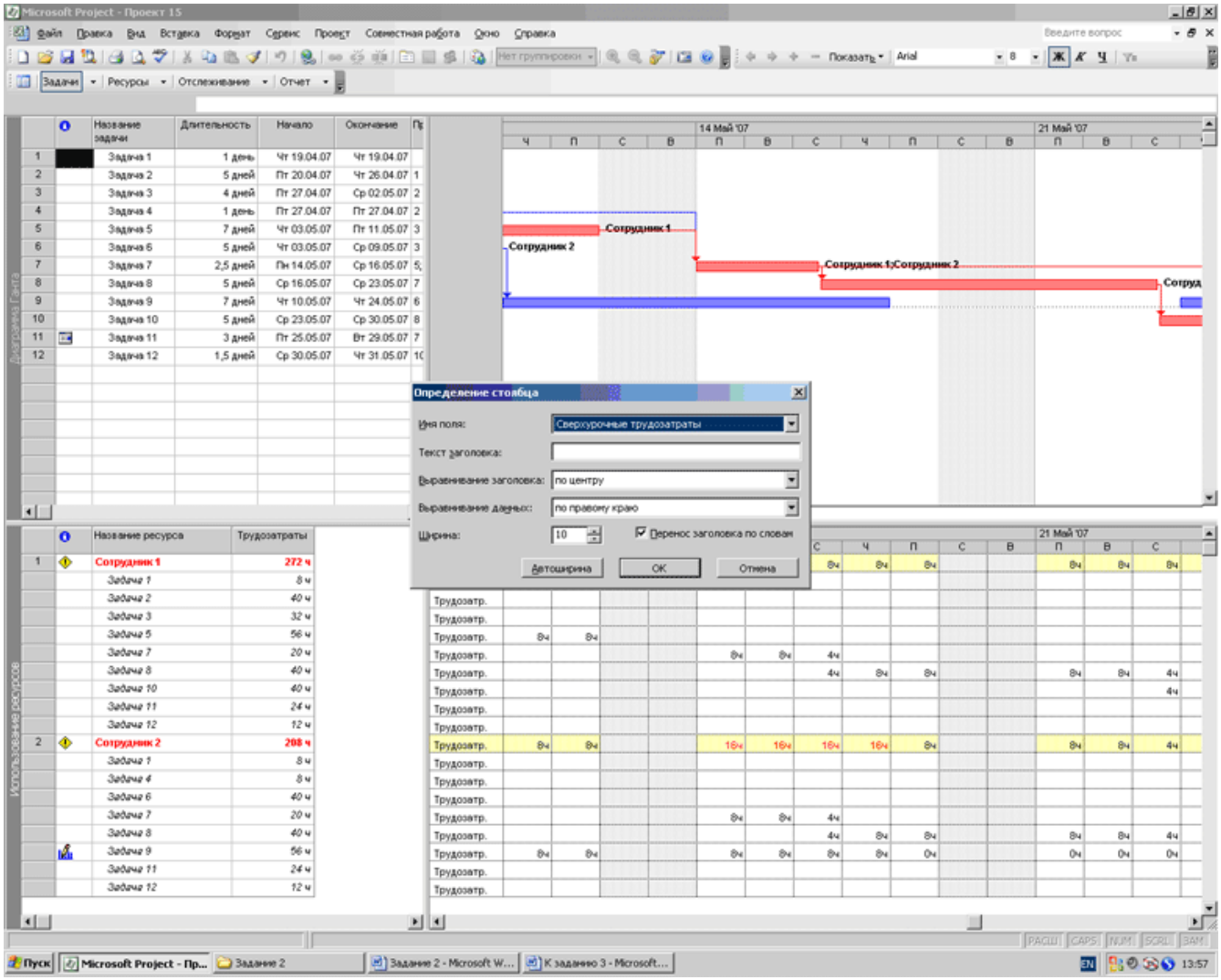


Назначение сверхурочных часов

Выбираем меню ***Вид/Использование ресурсов.***



Перегрузка отмечена красным цветом. Вставить новый столбец «Сверхурочные трудозатраты» (название выбирается из списка)



В таблице, которая после этого появляется, необходимо установить сверхурочные часы (трудозатраты) для конкретных задач.

Добавление нового сотрудника

Самостоятельно добавить в список ресурсов сотрудника 3 и назначить ему часть работ.

Все изменения графика перегрузки оформить в виде отчета.

Задание на самостоятельную работу

Требуется составить проект научно-исследовательской работы (НИР) с применением программы Microsoft Project и провести его оптимизацию.

В индивидуальное задание входят перечень и продолжительность работ, численность исполнителей. В научно-исследовательской работе принимают участие инженеры, научные сотрудники и лаборанты. Продолжительность работ задается как оптимистическая (минимальная), ожидаемая (средняя) и пессимистическая. Необходимо спроектировать ход работы, устранить возможные перегрузки.

Создать отчет, в котором:

1. Вывести на печать (до оптимизации):
    - сетевой график;
    - диаграмму Ганта;
    - таблицу: календарный план;
    - таблицу: суммарные данные;
    - список ресурсов;
    - графики загрузки ресурсов;
2. Вывести на печать (после оптимизации):
    - диаграмму Ганта;
    - таблицу: суммарные данные;

- графики загрузки ресурсов;
- отчет: сводка по проекту.

3. Смета затрат на разработку и реализацию проекта
4. Ответы на контрольные вопросы.
    1. Что такое смета проекта?
    2. Что такое прямые затраты (расходы), накладные (косвенные) затраты, общие и административные накладные расходы?
5. Выводы по проекту.

Итог работы: отчет

4. ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ПРАКТИЧЕСКИХ РАБОТ

4.1 Печатные изделия:

Основные:

О-1. Г.Н. Федорова. Разработка, администрирование и защита баз данных: Учебник для студ.учреждений сред. проф. образования/ Г.Н. Федорова. – 3-е изд., испр.- М.: Издательский центр «Академия», 2019. – 288 с.

О-2. Г.Н. Федорова. Разработка модулей программного обеспечения для компьютерных систем: Учебник для студ.учреждений сред. проф. образования/ Г.Н. Федорова. – 3-е изд., испр.- М.: Издательский центр «Академия», 2019. – 384 с.

О-3. Г.Н. Федорова. Осуществление интеграции программных модулей: Учебник для студ.учреждений сред. проф. образования/ Г.Н. Федорова. – 3-е изд., испр.- М.: Издательский центр «Академия», 2019. – 288 с.

Дополнительные:

Д-1. Волков Ю.И. Информационные системы: Учебник / Ю.И. Волков. - М.: Питер, 2006.

Д-2. Кокорева О.И., Реестр Windovs XP: / О.И. Кокорева - М.: БХВ-Перербург, 2008.

Д-3. Омельченко Л.Н., Федоров А.Ф., Реестр Windovs XP: самоучитель/ Л.Н. Омельченко, А.Ф. Федоров - М.: БХВ-Перербург, 2007.

Д-4. Голицына О.Л., Партыка Т.Л., Попов И.И. Программное обеспечение: учебное пособие/ О.Л. Голицына, Т.Л. Партыка, И.И. Попов - М.: ИД "ФОРУМ"-ИНФРА-М, 2006.

Д-5. Голицына О.Л., Партыка Т.Л., Попов И.И. Программное обеспечение: учебное пособие/ О.Л. Голицына, Т.Л. Партыка, И.И. Попов - М.: ИД "ФОРУМ"-ИНФРА-М, 2008.

Д-6. Ломов А.Ю. HTML, CSS, скрипты: практика создания сайтов / Ю.И. Волков. - М.: Питер, 2007.

Д-7. Титтел Э., Бурмейстер М. HTML для чайников/ Э.Титтел , М. Бурмейстер - М.: Вильямс, 2004.

Д-8. Полонская Е.Л., язык HTML: самоучитель/ Е.Л. Полонская - М.: Вильямс, 2005.

Д-9.Технология разработки программных продуктов: Практикум: учебник для студ. сред. проф. образования/ А. В. Рудаков, Федорова Г.Н. - 12-е изд., стер. – М.: Издательский центр «Академия– 208 стр. », 2017.

Д-10. Богданов В. В., Управление проектами в Microsoft Project 2007. Учебный курс, Уч. пособие, Издат. Питер, ISBN 978-5-469-00283-3, 592 стр., 2015 г..

4.2 Электронные издания (электронные ресурсы)

1. Единое окно доступа к информационным ресурсам [Электронный ресурс]. –Режим доступа: http://window.edu.ru/
2. *Федоров Г.Н., Учебник: Разработка модулей программного обеспечения для компьютерных систем, ИЦ Академия, 2017. - 350с., 25 подключений*
3. *Федоров Г.Н., Учебник: Осуществление интеграции программных модулей ИЦ Академия, 2017. - 282с., 25 подключений;*
4. *Федоров Г.Н., Учебник: Разработка, администрирование и защита баз данных ИЦ Академия, 2017. - 282с., 25 подключений;*

5. ЛИСТ ИЗМЕНЕНИЙ И ДОПОЛНЕНИЙ, ВНЕСЕННЫХ В МЕТОДИЧЕСКИЕ УКАЗАНИЯ

№ изменения, дата внесения, № страницы с изменением	
Было	**Стало**
Основание: **Подпись лица, внесшего изменения**	