Утверждено

на заседании каф. ПОКС

Протокол № \_\_\_\_ от \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Зав. каф. ПОКС \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тен И.Г.

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. И. РАЗЗАКОВА**

Кафедра «ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ КОМПЬЮТЕРНЫХ СИСТЕМ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# **ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ**

**КУРСОВОЙ ПРОЕКТ**

Методические указания для студентов специальности 710400

«Программная инженерия»

Профиль: Технология разработки программного обучения

дневной и заочной формы обучения

**Бишкек 2015**

**Аннотация**

Методические указания по курсовому проектированию по дисциплине «Объектно-ориентированное программирование» предназначены для студентов очной и заочной форм обучения специальности 710400 «Программная инженерия».

Приведено содержание курсового проекта, указан состав текстового и программного разделов проекта. Приведены варианты заданий для курсового проекта.

## 1. Цель курсового проекта

Курсовой проект по дисциплине «Объектно-ориентированное программирование» ставит своей целью закрепление и углубление навыков использования объектно-ориентированного подхода к программированию, полученных в процессе изучения дисциплины «Объектно-ориентированное программирование».

Изучение организации, разделяемых ресурсов, механизмов сообщений, использования динамически подключаемых библиотек, особенностей управления оперативной памятью в операционных системах Windows с точки зрения разработчика программного обеспечения; приобретение опыта создания прикладного программного обеспечения для Windows c использованием алгоритмического языка C++.

## 2. Предварительные требования

Требуется знание материала по следующим дисциплинам: «Информатика», «Программирование на языке высокого уровня», «Структуры и алгоритмы обработки данных».

## 3. Задачи курсового проекта

Для достижения поставленной выше цели, студентам предлагается, пользуясь средствами объектно-ориентированного языка С++, разработать программу. В результате выполнения данной работы студенты осваивают приемы практического использования объектно-ориентированного подхода в создании законченного программного продукта:

* реализующего выбранного в соответствии с вариантом задания,
* обладающего графическим интерфейсом пользователя, простого в установке и обслуживании.

## 4. Основные темы

1. Логические игры;
2. Управление базами данных;
3. Разработка реализации алгоритмов поиска;
4. Разработка реализации алгоритмов сортировки;
5. Разработать программу для расчета времени, необходимого для выполнения сортировок;
6. Разработка программ построения изображений фигур;
7. Разработка программ построения графиков.

## 5. Тематика и варианты заданий

## 5.1. Темы заданий

Вариант выбирается из утвержденного списка курсовых заданий по согласованию с преподавателем. Кроме того, вы можете самостоятельно придумать себе задание, отсутствующее в этом списке, предварительно согласовав его с преподавателем.

Требования для программ-игр: они должны предоставлять следующие возможности:

* начало игры на чистом поле, сброс предыдущей игры;
* выбор противника (человек, компьютер) (если применимо);
* отмену ходов;
* управление, как с клавиатуры так и мышью;
* контроль правильности ходов игрока(ов);
* определение конца игры;
* возможность задания произвольного размера поля (если применимо);
* возможность визуального редактирования уровней (если применимо);
* настройку клавиш управления;
* ведение списка чемпионов для каждого размера поля;
* выбор уровня сложности игры (если применимо).

**5.3. Методические указания**

Курсовой проект выполняется поэтапно, этапы приведены ниже в этом разделе.

**4.1. Написание технического задания**

Техническое задание (далее, ТЗ) содержит основные технические требования, предъявляемые к разрабатываемому программному обеспечению. В ТЗ указываются назначение программы, область ее применения, стадии разработки и т. д., а также особые требования, обусловленные спецификой конкретного программного обеспечения.

Правильно разработанное техническое задание позволяет студенту:

* представить готовую программу,
* выполнить попунктную проверку готовой программы (приемочное тестирование, проведение испытаний),
* уменьшить число ошибок, связанных с изменением требований в результате их неполноты или ошибочности (на всех стадиях и этапах создания, за исключением испытаний),
* спланировать выполнение проекта и работать по намеченному плану,

Со своей стороны, на основании ТЗ преподаватель может:

* убедиться, что студент правильно понял задание,
* требовать от студента соответствия программы всем условиям, оговоренным в ТЗ.

Техническое задание пишется в свободной форме и может, например, содержать такие разделы:

1. Обзор. Здесь можно рассмотреть назначение программы; требования к программе; ограничения, накладываемые на реализацию; выбрать платформу, язык программирования и т.п.
2. Модель программы. Здесь должно быть подробное словесное описание пользовательского интерфейса программы, ее поведения. Для большей наглядности могут быть приведены эскизы пользовательского интерфейса.
3. Архитектура программы. В этом разделе должна быть подробно расписана иерархия классов, используемых в программе, их интерфейсы и взаимодействие. Для большей наглядности можно использовать рисунки и диаграммы.
4. Форматы и протоколы. Здесь описываются форматы файлов, которые будут использованы в программе. Если программа будет иметь поддержку сети, то также определяется прикладной протокол взаимодействия клиента и сервера. Если программа будет использовать базу данных, ее структуру уместно так же привести в этом разделе. При написании этого пункта особое внимание необходимо обратить на существующие стандарты. Если для конкретной задачи уже существуют готовые решения, протоколы, форматы и т.п., являющиеся международными или национальными стандартами - лучше воспользоваться ими, а не придумывать собственное решение. Необходимо проведите исследование на предмет наличия таких стандартов, применительно к предметной области курсового проекта.
5. Тестирование. Для каждого разрабатываемого модуля программы должны быть приведены тесты, которыми будет проверяться корректность работы этого модуля.
6. План работы. Здесь должен быть по пунктам расписан порядок выполнения работы. Какие компоненты программы, в каком порядке будут реализованы и т.п.

### 4.2. Программная реализация

Написание и тестирование самой программы в соответствии с техническим заданием. Подробные требования к программной реализации приведены в разделе «[Организация выполнения и защиты курсового проекта](oop_termpaper.html#accounting)».

### 4.3. Документирование

Написание инструкций по сборке и установке программы, руководство пользователя, программиста. Подробные требования к документации приведены в разделе «[Организация выполнения и защиты курсового проекта](oop_termpaper.html#accounting)».

### 4.4. Защита

На защите студент представляет преподавателю программу, соответствующую техническому заданию, и документацию к программе. Преподаватель определяет степень соответствия представленной программы техническому заданию и аутентичность ее реализации.

## 5. Требования к программной реализации и документации

Основным требованием к программной реализации является обязательное использование объектно-ориентированного подхода при написании программы. Критериями использования объектно-ориентированного подхода являются использование:

* классов,
* механизма исключений,
* механизма наследования и множественного наследования,
* инкапсуляции,
* полиморфизма,

Помимо перечисленных выше базовых требований, программа, разрабатываемая в рамках курсового проекта должна удовлетворять ряду дополнительных требований, приведенных ниже.

### 5.1. Требования к среде разработки

Выбранная для реализации курсового проекта среда разработки должна удовлетворять следующим требованиям:

* поддержка современных 32-битных операционных систем,
* отсутствие лицензионных отчислений за право использовать систему (бесплатность),
* кроссплатформенность (возможность установить и использовать среду не только в Windows но и в UNIX-подобных ОС).

Таким образом, структурированные исходные коды курсового проекта облегчат преподавателю проверку и оценку работы.

### 5.3. Обеспечение переносимости на уровне исходного кода

Под переносимостью программного обеспечения на уровне исходного кода понимается возможность компиляции исходных кодов и сборки работоспособного исполняемого файла более чем на одной программной или аппаратной платформе. Например, переносимой можно считать программу, которая без модификации исходного кода компилируется и работает одновременно в операционных системах Windows и Linux, или Linux и FreeBSD, или FreeBSD и Symbian.

Переносимости на уровне исходных кодов можно добиться различными способами:

### 5.4. Оформление исходного кода

Исходный код программы необходимо оформлять правильно:

* как можно чаще пользуйтесь комментариями,
* не допускаются пустые строки,
* тщательно проверяйте параметры и возвращаемые значения на возможные ошибки.

В [[5](oop_termpaper.html#CLEARCODE)] даются советы, следование которым значительно облегчит отладку программного кода и его проверку преподавателем.

## 6. Организация выполнения и защиты курсового проекта

В качестве результата выполнения работы студент должен предоставить SVN репозиторий (для заочников), в котором будет содержаться:

1. техническое задание,
2. исходный код программы,
3. руководство пользователя,
4. руководство программиста,
5. отчет.

Кроме репозитория необходимо предоставить распечатанный отчет. Техническое задание пишется до написания программы. Это не значит, что пока не написано ТЗ, нельзя написать ни строчки программного кода. Конечно, вам придется экспериментировать, изучать возможности выбранного вами средства реализации, прежде чем вы сможете составить полноценное ТЗ. Важно только, что программа должна быть написана на основе технического задания и в соответствии с ним, а не техническое задание по уже готовой программе, как это иногда бывает.

Отчет является «вводным» документом. В отчете должна быть информация о структуре работы. Что это за работа (подробно), с чего начать проверку (по вашему мнению), каким образом развернуть исходные коды и скомпилировать программу.

Формально, за выполнение работы можно получить максимум 100 баллов. Оценке “отлично” соответствует 87-100 баллов. Оценке “хорошо” соответствует 74-86 баллов. Оценке “удовлетворительно” соответствует 60-73 баллам.

Баллы выставляются за каждый вовремя выполненный этап работы. Цена этапов следующая:

* черновик технического задания - 20 баллов,
* техническое задание - 20 баллов,
* программная реализация - 20 баллов,
* отладка, тестирование - 10 баллов,
* документация, защита - 30 баллов.

На каждый этап отводится время:

* черновик технического задания - 1 недели
* техническое задание - 1 неделя
* программная реализация и тестирование программной реализации - 1 неделя
* документация - 1 неделя
* защита -1 неделя

Для заочников результаты выполнения каждого этапа высылаются преподавателю по электронной почте. В случае невозможности высылать результаты по электронной почте в течение семестра, результаты каждого этапа, сразу после его выполнения, помещаются в SVN репозитарий и помечаются уникальной меткой. Наличие и верная дата установки метки в репозитории будет служить доказательством того, что этап выполнен вовремя.

Защита работы проводится в форме собеседования, во время сессии. Позднее выполнение этапа влечет за собой снижение балла. Небрежность в оформлении технического задания, документации, программного кода, влечет за собой снижение баллов на 10%-30%.

За все нестандартные решения (использование нового для вас объектно-ориентированного языка программирования; добавление в программу поддержки работы по сети и т.п.) будут начисляться дополнительные баллы. Количество зарабатываемых таким образом баллов оговаривается с преподавателем и назначается индивидуально.

**Литература**

1. Шилдт Г. "Программирование на С и С++ для Windows"
2. Р. Лафоре «Объектно-ориентированное программирование в С
3. Брюс Эккель — Философия С++. Введение в стандартный С++.
4. Брюс Эккель, Чак Эллисон — Философия C++. Практическое программирование.
5. Скотт Мэйерс — Эффективное использование C++. 55 верных советов улучшить структуру и код ваших программ.
6. Скотт Майерс — Эффективное использование С++. 35 новых способов улучшить стиль программирования.