Министерство науки и высшего образования РФ
Образовательная автономная некоммерческая организация
высшего образования
«Волжский университет имени В.Н. Татищева» (институт)

УТВЕРЖДАЮ
Ректор Якушин В.А.
от 02.05.2023г. № 77/1

Методическое указание по выполнению курсовой работы по дисциплине «Программирование»

Направление подготовки 09.03.01 Информатика и вычислительная техника

Квалификация (степень) выпускника – бакалавр

Форма обучения – очная, заочная, очно-заочная

Тольятти, 2023 г.

Методическое указание по выполнению курсовой работы по дисциплине «Программирование» составлено с требованиями ФГОС, ВО, ОПОП по направлению подготовки 09.03.01 Информатика и вычислительная техника (уровень высшего образования: бакалавриат) и учебного плана.

Методическое указание обсуждена и рекомендована к использованию и (или) изданию решением кафедры на заседании кафедры «Информатика и системы управления»
протокол № 09 от 19.04.2023г.
Зав. кафедрой ИиСУ
к.п.н., доцент Е.Н. Горбачевская

Одобрено Учебно-методическим советом вуза
протокол № 4/23 от 27.04.2023г
Председатель УМС
к.п.н. И.И. Муртаева

Курсовая работа по дисциплине «Программирование» ставит своей целью закрепление и углубление навыков использования объектно-ориентированного подхода к программированию, полученных в процессе изучения дисциплины «Программирование».

Содержание

I Общие требования к содержанию курсовой работы

Для достижения поставленной выше цели, пользуясь средствами объектно-ориентированного языка C++ или Python разработать программу – компьютерную логическую игру (по варианту). В результате выполнения данной работы студенты осваивают приемы практического использования объектно- ориентированного подхода в создании законченного программного продукта:

- реализующего выбранную (в соответствии с вариантом) компьютерную логическую игру,

- обладающего графическим интерфейсом пользователя,

- удовлетворяющего требованию переносимости на уровне исходного кода,

- простого в установке и обслуживании.

Декомпозиция системы на компоненты

Объектная методология предлагает рассматривать предметную область и проектировать программную систему как совокупность взаимодействующих друг с другом объектов. Объект обладает состоянием, поведением и идентичностью; структура и поведение схожих объектов определяет общий для них класс. Состояние объекта характеризуется перечнем всех свойств данного объекта и текущими значениями каждого из этих свойств. К числу свойств относятся присущие объекту или приобретаемые им характеристики, черты, качества или способности, делающие данный объект самим собой. Перечень свойств объекта является, как правило, статическим, поскольку эти свойства составляют неизменяемую основу объекта. Их принято называть атрибутами класса.

Поведение – это то, как объект действует и реагирует.

Поведение объекта определяется выполняемыми над ним операциями и его состоянием, причем некоторые операции имеют побочное действие: они изменяют состояние. Объектно-ориентированный стиль программирования

связан с воздействием на объекты путем передачиим сообщений (т.е. вызывая методы, описанные в их классе).

Операция над объектом порождает некоторую реакцию этого объекта.

Операции, которые можно выполнить по отношению к данному объекту, и реакция объекта на внешние воздействия определяют поведение этого объекта.

Операция – это услуга, которую может предоставить класс. На практике над объектами выполняются операции пяти видов.

Таблица 4 Операции над объектами

Конструктор	имеет то же имя, что и класс; определяет способ создания объекта или его инициализации
Деструктор	операция, выполняющая очистку памяти, когда объект класса выходит за пределы области видимости или он удаляется; имеет то же имя, что и класс со знаком «~» перед ним.
Модификатор	операция, которая изменяет состояние объекта
Селектор	операция, считывающая состояние объекта, но не меняющая состояния
Итератор	операция, позволяющая организовать доступ ко всем частям объекта в строго определенной последовательности

Объекты могут создаваться различными способами. Некоторые объекты являются локальными переменными, другие глобальными, третьи – членами классов и т.д.

Между объектами могут существовать различные отношения:

· ассоциация;

· наследование;

· агрегация;

· зависимость;

· и др.

Отношения двух любых объектов основываются на предположениях, которыми один обладает относительно другого: об операциях, которые можно выполнять и об ожидаемом поведении.

Связь – это специфическое сопоставление, через которое один объект (клиент) запрашивает услугу у другого объекта (сервера) или через которое один объект находит путь к другому. Она дает классу возможность узнавать об атрибутах, операциях и связях другого класса.

Ассоциация – это смысловая связь, которая не имеет направления и не объясняет, как классы общаются друг с другом.

Однако именно это требуется на ранней стадии анализа, поэтому мы фиксируем только участников, их роли и мощность отношения. В дальнейшем она, как правило, конкретизируется и принимает вид одного из рассматриваемых далее отношений.

Наследование – это такое отношение между классами, когда один класс повторяет структуру и поведение другого класса (одиночное наследование) или других (множественное наследование) классов. Класс, структура и поведение которого наследуются, называется суперклассом. Производный от суперкласса класс называется подклассом. Это означает, что наследование устанавливает между классами иерархию общего и частного.

Подкласс обычно расширяет или ограничивает существующую структуру и поведение своего суперкласса.

Связи наследования также называют обобщениями (generalization) и изображают в виде больших белых стрелок от класса- потомка к классу-предку.

Помимо наследуемых, каждый подкласс имеет свои собственные уникальные атрибуты, операции и связи.

Лучше всего строить иерархию классов, в которой от общих классов с помощью наследования образуются более специализированные.

Агрегация. В то время как связи обозначают равноправные или «клиент-серверные» отношения между объектами, агрегация описывает отношения целого и части, приводящие к соответствующей иерархии объектов, причем, идя от целого (агрегата) мы можем придти к его частям (атрибутам). В этом смысле агрегация – специализированный частный

случай ассоциации. Объект, являющийся атрибутом другого объекта (агрегата), имеет связь со своим агрегатом. Через эту связь агрегат может посылать ему сообщения.

Рассмотрим декомпозицию системы на компоненты на примере калькулятора, выполняющего операции с датами в виде дд.мм.гг.

Этот калькулятор выполняет сложение или вычитание двух дат и выдает результат в виде новой даты или количества дней (месяцев, недель), содержащихся в новой дате.

В этой системе можно выделить классы «Дата»,«Калькулятор» и «Диалог с пользователем».Объекты классов обладают свойствами, индивидуальностью и поведением. Свойства выражаются в виде атрибутов, индивидуальность – в виде значений соответствующих атрибутов, поведение - в виде методов, описанных в соответствующем классе.

Для выполнения арифметического действия Калькулятор должен знать две даты и знак операции. Следовательно, класс Калькулятор должен включать три объекта класса Дата (регистры): для операнда_1, операнда_2 и результата. Над операндами выполняется какая-то арифметическая операция, знак этой операции тоже можно хранить в отдельной переменной класса Калькулятор. Результат калькулятор выдает в виде одной из четырех форм: дата, количество дней, недель или месяцев. Таким образом, определили, что класс Калькулятор будет содержать следующие компонентные данные:

```
date op1,op2,rez;//операнды и результат
int znak;//код операции
int form;//код формы для вывода результата
```

Калькулятор необходимо инициализировать, следовательно, класс Калькулятор должен содержать конструктор. Кроме того, Калькулятор должен содержать модификаторы для изменения операндов, знака и формы вывода, селектор для получения результата и метод, который будет выполнять вычисление результата в зависимости от знака операции. Таким образом, получаем следующее описание класса Calc:

```
class Calc
{
Date op1;//операнд 1
Date op2;//операнд 2
Date res;//результат
int znak;//знак операции
int form;//форма вывода результата
public:
Calc();//конструктор
void set_op1(date);//модификатор для операнда 1
void set_op2(date); //модификатор для операнда 2
Date get_res();//селектор для получения результата
void set_znak(int); //модификатор для знака операции
void set_form(int); //модификатор для формы вывода
void execute();//вычисление результата
};
```

В классе Calc используются переменные типа Date – данные, с которыми работает Калькулятор. Между классами Calc и Date существует отношение агрегации, причем, это будет не просто агрегация, а композиция, т. к. операнды не могут существовать отдельно от калькулятора.

Рассмотрим свойства класса Дата. Над Датой можно выполнять операции, Дату надо вводить и выводить. Ввод и вывод Даты проще всего выполнять в виде строки. Но для выполнения операций с датами надо предусматривать представление Даты в виде чисел.

Операции по сложению и вычитанию дат можно выполнять по следующему алгоритму:

1. Перевести обе даты в дни.

2. Выполнить операцию.

3. Перевести результат в требуемую форму (дата, дни, недели, месяцы и т. п.)

Таким образом, класс Date должен иметь следующие компонентные данные:

int d;//количество дней в дате

char* s;//представление даты в виде строки

Дата вводится в форме строки, а затем вычисляется, сколько дней она содержит. Для преобразования Даты из одной формы в другую надо предусмотреть компонентные функции:

void days_to_string(); - переводит дни в строку

void string_to_days(); - переводит строку в дни

Над числами выполняются арифметические операции, следовательно, удобно будет перегрузить эти операции в виде компонентных операций-функций:

Date operator -(Date&D);

Date operator +(Date D);

Date& operator =(const Date &D);

Ввод и вывод чисел удобно реализовать с помощью глобальных перегруженных операций-функций >> и <<.

Таким образом, описание класса предназначенного для работы с датами будет выглядеть следующим образом:

```
class date
{
char *date_string;//строка для хранения даты в виде дд.мм.гггг.
int days;//количество дней в дате для выполнения расчетов
public:
date();//конструктор
date(const date&);//конструктор копирования
~date();//деструктор
//инициализация даты: заполняет атрибут date_string, вычисляет
//количество дней в этой дате, затем заполняет атрибут days
void init(char*);
```

```
// вспомогательная функция, которая по количеству дней формирует
//строку вида дд.мм.гггг.
void days_to_string();
// вспомогательная функция для вычисления дней в дате
void string_to_days();
//перегруженная функция для сложения двух дат
date operator+(date&);
//перегруженная функция для сложения двух дат
date operator-(date&);
//перегруженная функция присваивания
date& operator=(const date&);
//дружественная функция для ввода даты
friend istream&operator>>(istream&in,date&);
//дружественная функция для вывода даты
friend ostream&operator<<(ostream&in,date&);
};
```

Класс «Диалог с пользователем» осуществляет ввод данных в виде строк, заносит эти строки в операнды калькулятора, вводит операцию, заносит ее в знак операции калькулятора, вводит форму, требуемого результата, заносит ее в соответствующий атрибут калькулятора, получает результат и выводит его в нужной форме. Эти действия повторяются до тех пор, пока пользователь не введет команду для окончания расчетов. Для ввода данных можно использовать меню. Таким образом, класс Диалог может иметь следующую реализацию:

```
class Dialog
{
public:
Calc c;//калькулятор для выполнения расчетов
void menu();//меню для диалога с пользователем
//цикл, в котором осуществляется вывод меню, ввод данных,
```

```
//вычисление результата и проверка окончания расчетов
void doing();
};
```

Основная функция будет иметь вид:

```
void main()
{
Dialog d;
d.doing();
}
```

II Содержание разделов пояснительной записки

1. Курсовая работа выполняется в среде DEV C++ как консольное приложение.

2. При выполнении курсовой работы обязательным является использование объектно-ориентированного программирования.

3. Как правило, класс как тип, определенный пользователем, должен содержать скрытые поля и следующие функции

– Конструкторы, определяющие, как инициализируются объекты класса;

– Набор методов, реализующих свойства класса (методы, возвращающие значения скрытых полей класса описываются с модификатором const, для того, чтобы не изменялись значения полей);

– Набор операций, позволяющий копировать, присваивать, сравнивать объекты и производить с ними требуемые действия;

– Класс исключений, используемый для сообщений об ошибках с помощью генерации исключительных ситуаций.

4. В курсовой работе должно использоваться не менее 3 классов (см. п.4), причем диалог с пользователем должен быть реализован как отдельный класс.

5. Каждый класс должен быть реализован в виде двух файлов:

заголовочного (.h), содержащего описание класса и файла (.cpp), содержащего реализацию методов класса. Основная функция main реализуется в виде отдельного файла. Если в работе используются глобальные функции, они также должны быть размещены в отдельном файле.

6. В курсовой работе должны использоваться перегруженные функции-операции для выполнения заданных в варианте операций.

Например, для добавления элемента в список можно перегрузить операцию сложения (+) или инкремент (++).

7. Для реализации протокола (варианты 1-15) и а также записи данных в файл и получения данных из файла (варианты 16-25) использовать файловые потоки.

8. Предусмотреть проверку корректности данных. При проверке использовать механизм исключительных ситуаций.

Рекомендации по программированию

· При создании класса следует хорошо продумать его интерфейс – средства работы с классом для тех программ, которые будут его использовать. Интерфейс должен быть интуитивно понятным и включать только методы. Поля данных класса должны быть скрытыми.

· Не следует определять методы типа get/set для всех скрытых полей класса — это все равно, что открыть к ним доступ, только более сложным способом. Поля класса вводятся только для того, чтобы реализовать свойства класса, представленные в его интерфейсе с помощью методов.

· Не нужно расширять интерфейс класса без необходимости, «на всякий случай», поскольку увеличение количества методов ведет к трудности понимания класса пользователем. В идеале интерфейс должен быть полным, то есть предоставлять возможность выполнить любые разумные действия с классом, и минимальным — без дублирования и пересечения возможностей методов.

· В виде методов рекомендуется определять только действия,

реализующие свойства класса. Если какое-либо действие можно реализовать, не обращаясь к скрытым полям класса, его нет необходимости описывать как метод; лучше описать его как обычную функцию, поместив ее в общее с классом пространство имен. Если функция выполняет действие, не являющееся свойством класса, но нуждается в доступе к его скрытым полям, ее следует объявить как дружественную. Но в общем случае дружественных функций и классов надо избегать, поскольку главной идеей ООП является минимизация связей между инкапсулированными классами.

· Для увеличения производительности программы наиболее часто вызываемые методы можно объявить как встроенные (inline). В основном это касается коротких методов, тело которыхоказывается меньше размера кода, генерируемого для их вызова.

Кроме ускорения программы за счет исключения вызовов, это дает возможность компилятору производить более полную оптимизацию.

Однако необходимо учитывать, что директива inline носит для компилятора рекомендательный характер.

· Конструкторы и деструкторы делать встраиваемыми не рекомендуется, поскольку в них фактически присутствует код, помещаемый компилятором, размер этого кода может быть весьма значительным (например, в конструкторе производного класса должны быть вызваны конструкторы всех базовых и вложенных классов).

· Перегруженные операции класса должны иметь интуитивно понятный общепринятый смысл (например, не следует заставлять операцию + выполнять что-либо, кроме сложения или добавления).

· Если какая-либо операция перегружена, следует, если возможно, перегрузить и аналогичные операции, например, +, += и ++ (компилятор этого автоматически не сделает). При этом операции должны иметь ту же семантику, что и их стандартные аналоги.

· И конструктор копирования, и операция присваивания, создаваемые по умолчанию, выполняют поэлементное копирование из области-источника

в область-приемник. Если объект содержит указатели, это приведет к тому, что после копирования два соответствующих указателя разных объектов будут ссылаться на одну и ту же область памяти. При уничтожении первого из объектов эта память будет освобождена, а повторная попытка освободить ее при уничтожении второго объекта приведет к неопределенному поведению программы. Поэтому для классов, содержащих поля-указатели, следует всегда явно определять конструктор копирования и операцию присваивания, выполняющие выделение памяти под динамические поля объекта.

· Динамическая память, выделенная в конструкторе объекта, должна освобождаться в его деструкторе. Невыполнение этого требованияприводит к утечкам памяти. Удаление нулевого указателя безопасно (при этом ничего не происходит), поэтому если конструкторы, конструкторы копирования и операция присваивания написаны правильно, любой указатель либо ссылается на выделенную область памяти, либо равен нулю, и к нему можно применять delete без проверки.

· Разница между конструктором копирования и операцией присваивания заключается в том, что последняя работает в том случае, когда объект-приемник уже существует, поэтому в ней перед выделением динамической памяти следует освободить память занятую ранее. Из этого следует, что при реализации операции присваивания для классов, содержащих поля-указатели,

необходимо проводить проверку на самоприсваивание и в этом случае оставить объект без изменений. Необходимо также помнить о том, что операция присваивания должна возвращать ссылку на константу.

· В конструкторах для задания начальных значений полям рекомендуется использовать инициализацию, а не присваивание.

Инициализация более универсальна, так как может применяться в тех случаях, когда присваиванием пользоваться нельзя (например, при задании значений константным полям или ссылкам). Кроме того, она выполняется

более эффективно, потому что создание объекта в C++ начинается с инициализации его по-лей конструктором по умолчанию, после чего выполняется вызываемый конструктор. Необходимо учитывать и тот факт, что поля инициализируются в порядке их объявления, а не в порядке появления в списке инициализации. Поэтому для уменьшения числа возможных ошибок порядок указания полей в списке инициализации конструктора должен соответствовать порядку их объявления в классе.

· Статические поля не должны инициализироваться в конструкторе, поскольку им нужно присваивать начальное значение только один раз для каждого класса, а конструктор выполняется для каждого объекта класса. Статические поля инициализируются в глобальной области определения (вне любой функции).

· Конструкторы копирования также должны использовать списки инициализации полей, поскольку иначе для базовых классов и вложенных объектов будут вызваны конструкторы по умолчанию.

· Операция присваивания не наследуется, поэтому она должна быть определена в производных классах. При этом из нее следует явным образом вызывать соответствующую операцию базового класса

· Открытое наследование класса Y из класса X означает, что Y представляет собой разновидность класса X, то есть более конкретную, частную концепцию. Базовый класс X является более общим понятием, чем Y. Везде, где можно использовать X, можно использовать и Y, но не наоборот (вспомните, что на место ссылок на базовый класс можно передавать ссылку на любой из производных). Необходимо помнить, что во время выполнения программы не существует иерархии классов и передачи сообщений объектам базового класса из производных — есть только конкретные объекты классов, поля которых формируются на основе иерархии на этапе компиляции.

· Методы, которые должны иметь все производные классы, но которые не могут быть реализованы на уровне базового класса, должны быть

виртуальными. Например, все объекты иерархии должны уметь выводить информацию о себе. Поскольку она хранится в различных полях производных классов, эту функцию нельзя реализовать в базовом классе. Естественно назвать ее во всех классах одинаково и объявить как виртуальную с тем, чтобы другие методы базового класса могли вызывать ее в зависимости от фактического типа объекта, с которым они работают. По этой причине деструкторы объявляются как виртуальные.

· При переопределении виртуальных методов нельзя изменять наследуемое значение аргумента по умолчанию, поскольку по правилам C++ оно определяется типом указателя, а не фактическим типом объекта, вызвавшего метод.

Требования к оформлению исходных кодов и документации

Для контроля за процессом написания программы, возможности отката к предыдущим версиям, возможности совместной работы над программным кодом все исходные коды программы и вся сопутствующая документация (ТЗ, отчет и т.п.) должны храниться в репозитарии под управлением системы контроля версий **Subversion** [1] (далее, SVN).

Документацию по системе **Subversion** можно найти в [2].

Оффлайн-версию книги можно получить у преподавателя.

Перед защитой выполненной курсовой работы преподавателю сдается весь SVN репозитарий (а не рабочая копия работы!), упакованный в архив формата ZIP.

На дерево исходных кодов не налагается никаких специальных требований, вы можете организовывать их так, как вам удобно. Однако, за многие годы сообщество разработчиков программного обеспечения выработало более менее универсальные подходы к структурированию дерева каталогов.

Рассмотрим пример. Пусть, мы разрабатываем программу "Морской бой". Логично назвать SVN модуль (а значит и корневой каталог исходных кодов программы) по названию программы:

seabattle. Внутри этого каталога могли бы размещаться такие файлы и каталоги:

README

Файл (в формате ASCII), содержащий краткое описание программы и указания, где искать подробную документацию.

AUTHORS

Файл, содержащий список авторов программы. Вы могли бы указать здесь ваше имя, номер группы, почтовый адрес, вклад в программу (если разработчиков несколько).

INSTALL

Файл, содержащий инструкции по компиляции и инсталляции программы в систему.

TODO

В этом файле можно хранить список обнаруженных ошибок, которые планируется исправить в будущем, или нереализованных еще возможностей и т.п.

ChangeLog

В этом файле ведется хронология официальных версий программы, с указанием даты выхода очередной версии, ее номера, списка исправленных ошибок и улучшений для каждой версии.

COPYRIGHT

Здесь обычно приводится лицензия, под которой распространяются исходные коды программы.

VERSION

Последняя выпущенная версия программы. Этот файл обычно используют при автоматической сборке дистрибутива программы, чтобы вставить номер версии в имя файла дистрибутива и т.п.

src

Каталог, в котором находятся исходные тексты программы.

doc

В этом каталоге можно разместить документация на программу: техническое задание, руководство пользователя и т.п.

contrib

Здесь размещают сторонние модули. Библиотеки сторонних разработчиков, которые вы использовали в своей работе и т.п.

share

В этом каталоге можно поместить различные системно-независимые файлы: иконки, спрайты, игровые уровни, карты и т.п.

Таким образом структурированные исходные коды курсовой работы облегчат преподавателю проверку и оценку работы.

Оформление исходного кода

Исходный код программы необходимо оформлять правильно:

· как можно чаще пользуйтесь комментариями;

· выносите константы в #define;

· тщательно проверяйте параметры и возвращаемые значения на возможные ошибки.

В [5] даются советы, следование которым значительно облегчит отладку программного кода и его проверку преподавателем.

Лицензионная чистота реализации

Все средства разработки, использованные в курсовой работе (компилятор, среда выполнения, сторонние библиотеки и т.д.) должны позволять использовать их без лицензионных отчислений. Другими словами, преподаватель должен иметь возможность установить на своем компьютере все необходимое программное обеспечение, не покупая лицензий и не нарушая действующее законодательство.

Обратите внимание, что не только средства реализации, но и средства оформления программного продукта являются интеллектуальной собственностью, защищенной законодательством.

Используемые в программе изображения (иконки, логотипы и т.п.) тоже должны быть лицензионно чистыми. Для оформления

пользовательского интерфейса можно воспользоваться какой либо свободной библиотекой изображений. Мы рекомендуем библиотеку «**Tango!**» [6].

III Типовые темы курсовой работы

КОМПЛЕКТ №1 - КЛАССЫ

Тема 1. Разработка класса для представления множества целых чисел на основе связанного списка. Операции – включение элемента, исключение элемента, поиск элемента, объединение множеств, пересечение множеств.

Тема 2. Разработка класса для представления многочлена от одной переменной на основе связанного списка. Операции: сложение многочленов, умножение многочленов, деление многочленов, дифференцирование, интегрирование.

Тема 3. Разработка класса для представления упорядоченного множества вещественных чисел на основе циклического связанного списка. Операции – включение элемента, исключение элемента, поиск элемента, объединение множеств, пересечение множеств.

Тема 4. Разработка класса для представления многочлена от одной переменной на основе двунаправленного связанного списка. Операции: сложение многочленов, умножение многочленов, деление многочленов, дифференцирование, интегрирование.

Тема 5. Разработка класса для представления упорядоченного множества строк на основе бинарного дерева. Операции – включение элемента, исключение элемента, поиск элемента, объединение множеств, пересечение множеств.

Тема 6. Разработка класса для представления многочлена от одной переменной на основе циклического связанного списка. Операции: сложение многочленов, умножение многочленов, деление многочленов, дифференцирование, интегрирование.

Тема 7. Разработка класса для представления множества целых чисел

на основе хеширования со связанными цепочками. Операции – включение элемента, исключение элемента, поиск элемента, объединение множеств, пересечение множеств.

Тема 8. Разработка класса для представления многочлена от двух переменных на основе связанного списка. Операции: сложение многочленов, умножение многочленов.

Тема 9. Разработка класса для представления множества целых чисел на основе хеширования в таблице. Операции – включение элемента, исключение элемента, поиск элемента, объединение множеств, пересечение множеств.

Тема 10. Разработка класса для представления множества вещественных чисел на основе двунаправленного связанного списка. Операции – включение элемента, исключение элемента, поиск элемента, объединение множеств, пересечение множеств.

Тема 11. Создать шаблон класса массив с методом сортировки, отсортировать свой класс “Адрес” по его полям город, улица, индекс.

Тема 12. Создать шаблон циклической очереди. С помощью него обработать ввод с клавиатуры, заполнение информацией вашей памяти.

Тема 13. Создать шаблон приоритетной очереди. При добавлении элемента в такую очередь порядковый номер нового элемента определяется его приоритетом..

Тема 14. С помощью шаблона класса стек написать программу калькулятор с операциями сложения, вычитания, деления, умножения, возведения в степень.

Тема 15. Создать шаблон связный список с двойными ссылками, в котором просмотр списка в любом направлении, поиск в списке конкретного элемента, удаление элемента из списка, ввод нового элемента в список, получение указателя на начало и конец списка, ввод нового элемента в список.

Тема 16. Создать шаблон бинарное дерево, в котором можно

проходить по дереву одним из трех способов: последовательно, нисходящим и восходящим. Добавлять элементы в дерево, поиск по дереву, удалять элементы из дерева.

Базовый класс будет определять природу объектов, формирующих дерево, а от него наследовать класс, определяющий операции на деревом.

Тема 17. Описать класс, реализующий стек. Написать программу, использующую этот класс для моделирования Т-образного сортировочного узла на железной дороге. Программа должна разделять на два направления состав, состоящий из вагонов двух типов (на каждое направление формируется состав из вагонов одного типа). Предусмотреть возможность формирования состава из файла и с клавиатуры.

Тема 18. Описать класс, реализующий бинарное дерево, обладающее возможностью добавления новых элементов, удаления существующих, поиска элемента по ключу, а также последовательного доступа ко всем элементам.

Написать программу, использующую этот класс для представления англо-русского словаря. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса. Предусмотреть возможность формирования словаря из файла и с клавиатуры.

Тема 19. Построить систему классов для описания плоских геометрических фигур: круга, квадрата, прямоугольника. Предусмотреть методы для создания объектов, перемещения на плоскости, изменения размеров и вращения на заданный угол.

Написать программу, демонстрирующую работу с этими классами. Программа должна содержать меню, позволяющее осуществить проверку всех методов классов.

Тема 20. Построить описание класса, содержащего информацию о почтовом адресе организации. Предусмотреть возможность раздельного изменения составных частей адреса, создания и уничтожения объектов этого класса.

Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

КОМПЛЕКТ №2 - Использование стандартной библиотеки шаблонов (STL)

Тема 1. Программа моделирования Т-образного сортировочного узла на железной дороге с использованием контейнерного класса stack из STL.

Программа должна разделять на два направления состав, состоящий из вагонов двух типов (на каждое направление формируется состав из вагонов одного типа). Предусмотреть возможность ввода исходных данных с клавиатуры и из файла.

Тема 2. Программа моделирования работы автобусного парка.

Сведения о каждом автобусе содержат: номер автобуса, фамилию и инициалы водителя, номер маршрута.

Программа должна обеспечивать выбор с помощью меню и выполнение одной из следующих функций:

– начальное формирование данных о всех автобусах в парке в виде списка (ввод с клавиатуры или из файла);

– имитация выезда автобуса из парка: вводится номер автобуса; программа удаляет данные об этом автобусе из списка автобусов, находящихся в парке, и записывает эти данные в список автобусов, находящихся на маршруте;

– имитация въезда автобуса в парк: вводится номер автобуса; программа удаляет данные об этом автобусе из списка автобусов, находящихся на маршруте, и записывает эти данные в список автобусов, находящихся в парке;

– вывод сведений об автобусах, находящихся в парке, и об автобусах, находящихся на маршруте.

Для представления необходимых списков использовать контейнерный класс 1ist.

Тема 3. Написать программу учета заявок на авиабилеты.

Каждая заявка содержит: пункт назначения, номер рейса, фамилию и инициалы пассажира, желаемую дату вылета.

Программа должна обеспечивать выбор с помощью меню и выполнение одной из следующих функций:

– добавление заявок в список;

– удаление заявок;

– вывод заявок по заданному номеру рейса и дате вылета;

– вывод всех заявок.

Для хранения данных использовать контейнерный класс 1ist.

Тема 4. Написать программу учета книг в библиотеке.

Сведения о книгах содержат: фамилию и инициалы автора, название, год издания, количество экземпляров данной книги в библиотеке.

Программа должна обеспечивать выбор с помощью меню и выполнение одной из следующих функций:

– добавление данных о книгах, вновь поступающих в библиотеку;

– удаление данных о списываемых книгах;

– выдача сведений о всех книгах, упорядоченных по фамилиям авторов;

– выдача сведений о всех книгах, упорядоченных по годам издания.

Хранение данных организовать с применением контейнерного класса multimap, в качестве ключа использовать «фамилию и инициалы автора».

Тема 5. Программа «Моя записная книжка».

Предусмотреть возможность работы с произвольным числом записей, поиска записи по какому-либо признаку (например, по фамилии, дате рождения или номеру телефона), добавления и удаления записей, сортировки по разным полям.

Хранение данных организовать с применением контейнерного класса шар или multimap'

Тема 6. Программа «Маклер» для учета заявок на обмен квартир и поиска вариантов обмена.

Каждая заявка содержит сведения о двух квартирах: требуемой

(искомой) и имеющейся. Сведения о каждой квартире содержат: количество комнат, площадь, этаж, район.

Программа должна обеспечивать выбор с помощью меню и выполнение одной из следующих функций:

– ввод заявки на обмен;

– поиск в картотеке подходящего варианта: при совпадении требований и предложений по количеству комнат и этажности и различии по показателю «площадь» в пределах 10% выводится соответствующая карточка и удаляется из списка, в противном случае поступившая заявка включается в картотеку;

– вывод всей картотеки.

Для хранения данных картотеки использовать контейнерный класс 1ist.

Тема 7. Программа «Автоматизированная информационная система на железнодорожном вокзале».

Информационная система содержит сведения об отправлении поездов дальнего. следования. Для каждого поезда указывается: номер поезда, станция назначения, время отправления.

Программа должна обеспечивать выбор с помощью меню и выполнение одной из следующих функций:

– первоначальный ввод данных в информационную систему (с клавиатуры или из файла);

– вывод сведений по всем поездам;

– вывод сведений по поезду с запрошенным номером;

– вывод сведений по тем поездам, которые следуют до запрошенной станции назначения.

Хранение данных организовать с применением контейнерного класса vector.

Тема 8. Программа «Англо-русский и русско-английский словарь».

«База данных» словаря должна содержать синонимичные перевода слов. Программа должна обеспечивать выбор с помощью меню и выполнение

одной из следующих функций:

- Загрузка «базы данных» словаря (из файла).
- Выбор режима работы: англо-русский или русско-английский.
- Вывод перевода заданного английского слова.
- Вывод перевода заданного русского слова.

Базу данных словаря реализовать в виде двух контейнеров типа tap.

Тема 9. Программа, реализующую игру «Крестики-нолики» между двумя игроками: пользователем и компьютером (роботом). В программе использовать контейнерные классы STL.

Тема 10. Логическая игра-головоломка «Игра в 15». Начальное размещение номеров — случайное. Предусмотреть два режима демонстрации решения: непрерывный (с некоторой задержкой визуализации) и пошаговый (по нажатию любой клавиши). В программе использовать контейнерные классы STL.

Тема 11. Программа формирования списка кандидатов, участвующих в выборах мэра.

Каждая заявка от кандидата содержит: фамилию и инициалы, дату рождения, место рождения, индекс популярности.

Программа должна обеспечивать выбор с помощью меню и выполнение одной из следующих функций:

1.Добавление заявки в список кандидатов. Для ввода индекса популярности (значение указано в скобках) предусмотреть выбор с помощью подменю одного из следующих вариантов:

поддержан президентом (70);

поддержан оппозиционной партией (15);

оппозиционный кандидат, который снимет свою кандидатуру в пользу кандидата № 1 (10);

прочие (5).

2.Удаление заявки по заявлению кандидата.

3.Формирование и вывод списка для голосования.

Хранение данных организовать с применением контейнерного класса priority_ queue из STL. Для надлежащего функционирования очереди с приоритетами побеспокоиться о надлежащем определении операции < (меньше) в классе, описывающем заявку кандидата. Формирование и вывод списка для голосования реализовать посредством выборки заявок из очереди.

Тема 12. Программа моделирования работы автобусного парка.

Сведения о каждом автобусе содержат: номер автобуса, фамилию и инициалы водителя, номер маршрута.

Программа должна обеспечивать выбор с помощью меню и выполнение одной из следующих функций:

– начальное формирование данных о всех автобусах в парке в виде списка (ввод с клавиатуры или из файла);

– имитация выезда автобуса из парка: вводится номер автобуса; программа удаляет данные об этом автобусе из списка автобусов, находящихся в парке, и записывает эти данные в список автобусов, находящихся на маршруте;

– имитация въезда автобуса в парк: вводится номер автобуса; программа удаляет данные об этом автобусе из списка автобусов, находящихся на маршруте, и записывает эти данные в список автобусов, находящихся в парке;

– вывод сведений об автобусах, находящихся в парке, и об автобусах, находящихся на маршруте, упорядоченных по номерам автобусов;

– вывод сведений об автобусах, находящихся в парке, и об автобусах, находящихся на маршруте, упорядоченных по номерам маршрутов.

Хранение всех необходимых списков организовать с применением контейнерного класса map, в качестве ключа использовать «номер автобуса».

Тема 13. Программа учета заявок на авиабилеты.

Каждая заявка содержит: пункт назначения, номер рейса, фамилию и инициалы пассажира, желаемую дату вылета.

Программа должна обеспечивать выбор с помощью меню и

выполнение одной из следующих функций:

- добавление заявок в список;
- удаление заявок;
- вывод заявок по заданному номеру рейса и дате вылета;
- вывод всех заявок, упорядоченных по пунктам назначения;
- вывод всех заявок, упорядоченных по датам вылета.

Хранение данных организовать с применением контейнерного класса multimap, в качестве ключа использовать «пункт назначения».

Тема 14. Программа учета книг в библиотеке.

Сведения о книгах содержат: фамилию и инициалы автора, название, год издания, количество экземпляров данной книги в библиотеке.

Программа должна обеспечивать выбор с помощью меню и выполнение одной из следующих функций:

- добавление данных о книгах, вновь поступающих в библиотеку;
- удаление данных 6 списываемых книгах;
- выдача сведений о всех книгах, упорядоченных по фамилиям авторов;
- выдача сведений о всех книгах, упорядоченных по годам издания.

Хранение данных организовать с применением контейнерного класса vector.

Тема 15. Программа учета заявок на обмен квартир и поиска вариантов обмена.

Каждая заявка содержит фамилию и инициалы заявителя, а также сведения о двух квартирах: требуемой (искомой) и имеющейся. Сведение о каждой квартире содержат: количество комнат, площадь, этаж, район.

Программа должна обеспечивать выбор с помощью меню и выполнение одной из следующих функций:

- ввод заявки на обмен;
- поиск в картотеке подходящего варианта: при совпадении требований

и предложений по количеству комнат и этажности и различии по показателю «площадь» в пределах 10% выводится соответствующая карточка и удаляется из списка, в противном случае поступившая заявка включается в картотеку;

– вывод всей картотеки.

Хранение данных организовать с применением контейнерного класса set.

Тема 16. Программа «Автоматизированная информационная система на железнодорожном вокзале».

Информационная система содержит сведения об отправлении поездов дальнего следования. Для каждого поезда указывается: номер, станция назначения, время отправления.

Программа должна обеспечивать выбор с помощью меню и выполнение одной из следующих функций:

– первоначальный ввод данных в информационную систему (с клавиатуры или из файла);

– вывод сведений по всем поездам;

– вывод сведений по поезду с запрошенным номером;

– вывод сведений по тем поездам, которые следуют до запрошенной станции назначения.

Хранение данных организовать с применением контейнерного класса set.

Тема 17. Программа «Англо-русский и русско-английский словарь».

«База данных» словаря должна содержать синонимичные варианты перевода слов. Программа должна обеспечивать выбор с помощью меню и выполнение одной из следующих функций:

– Загрузка «базы данных» словаря (из файла).

– Выбор режима работы: (англо-русский; русско-английский)

– Вывод вариантов перевода заданного английского слова.

– Вывод вариантов перевода заданного русского слова.

КОМПЛЕКТ №3 – Игровые программы

Тем не менее, все программы должны предоставлять следующие возможности:

· начало игры на чистом поле, сброс предыдущей игры;

· выбор противника (человек, компьютер) (если применимо);

· сохранение текущей игры в любой момент игры в файл, восстановление состояния игры из файла;

· отмену ходов;

· подсказку следующего хода;

· контроль правильности ходов игрока(ов);

· определение конца игры, отслеживание патовых ситуаций;

· возможность задания произвольного размера поля (если применимо);

· возможность визуального редактирования уровней (если применимо);

· управление как с клавиатуры так и мышью;

· настройку клавиш управления;

· использование горячих клавиш;

· возможность задания настроек в конфигурационном файле или реестре Windows;

· индикацию текущего счета (если применимо);

· ведение списка чемпионов для каждого размера поля;

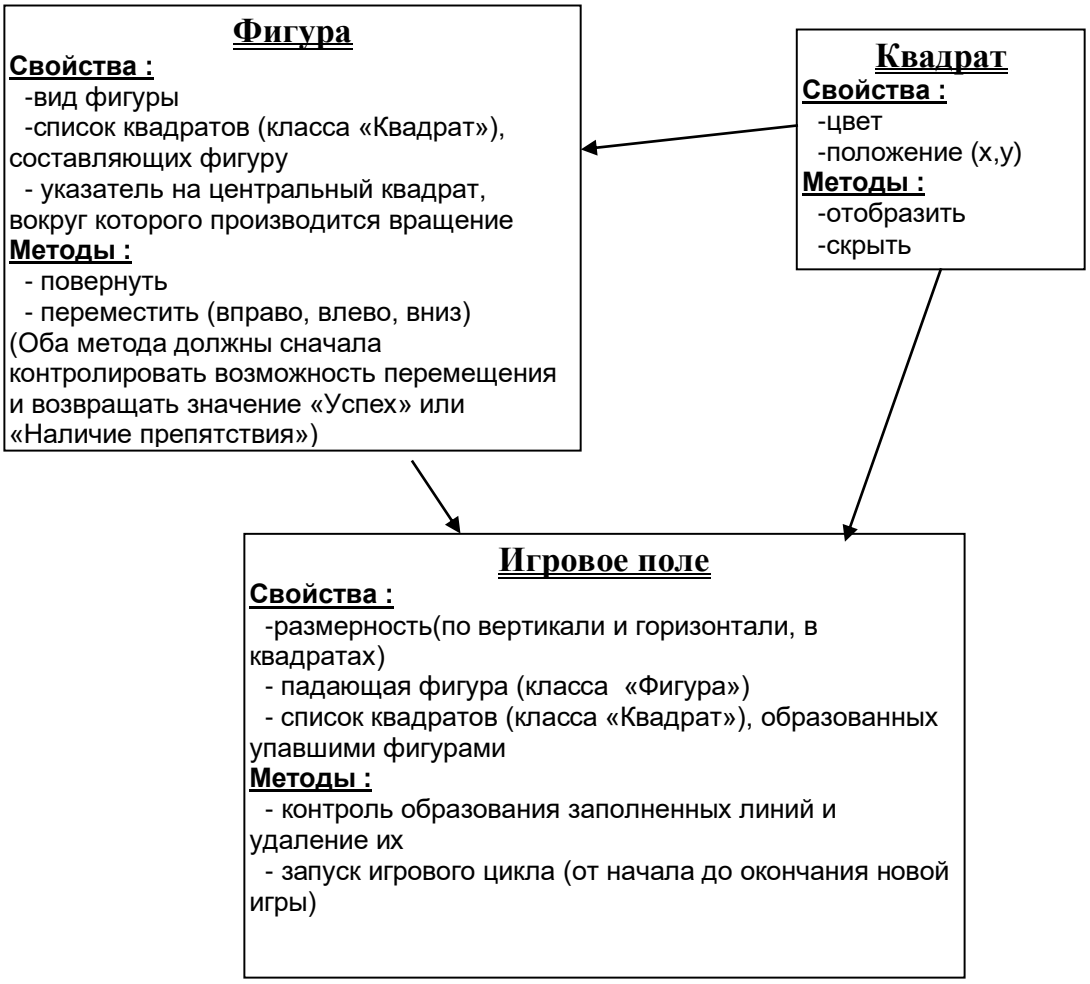
· выбор уровня сложности игры (если применимо)

· вынесение графических элементов (курсоры, иконки, спрайты) в ресурсный файл (если применимо)

· наличие инсталлятора (показ лицензии, выбор устанавливаемых компонентов, выбор пути установки и т.д.)

Тема 1. Игра “Тетрис”.

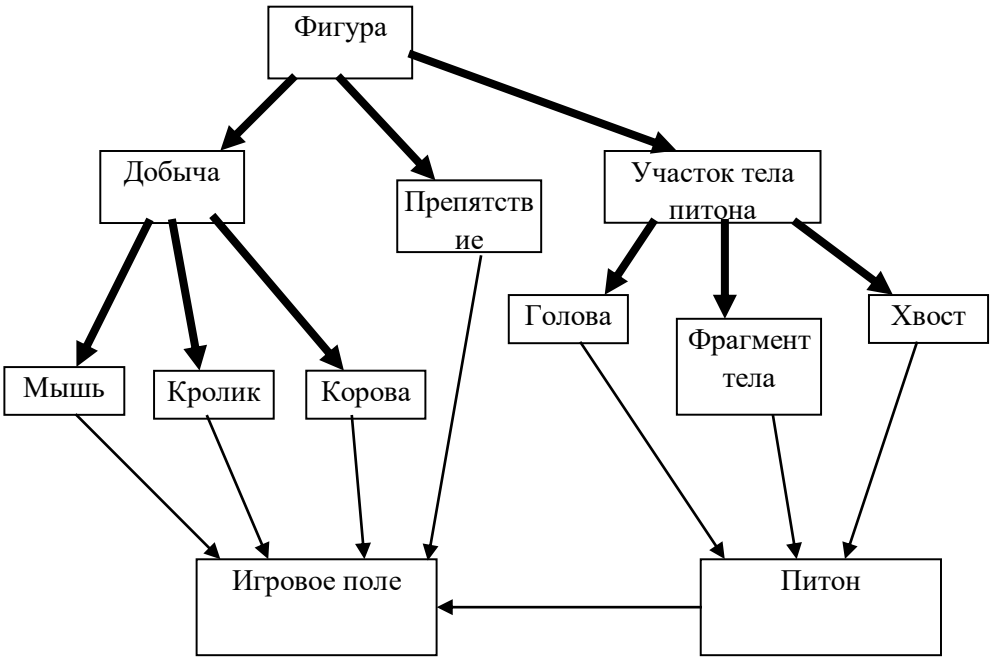
Создать классы, соответствующие следующей иерархии:



В начале падения новой фигуры объект класса «Фигура» должен создаваться, а при окончании падения этот объект должен уничтожаться, а принадлежащие ему квадраты должны «передаваться» объекту «Игровое поле».

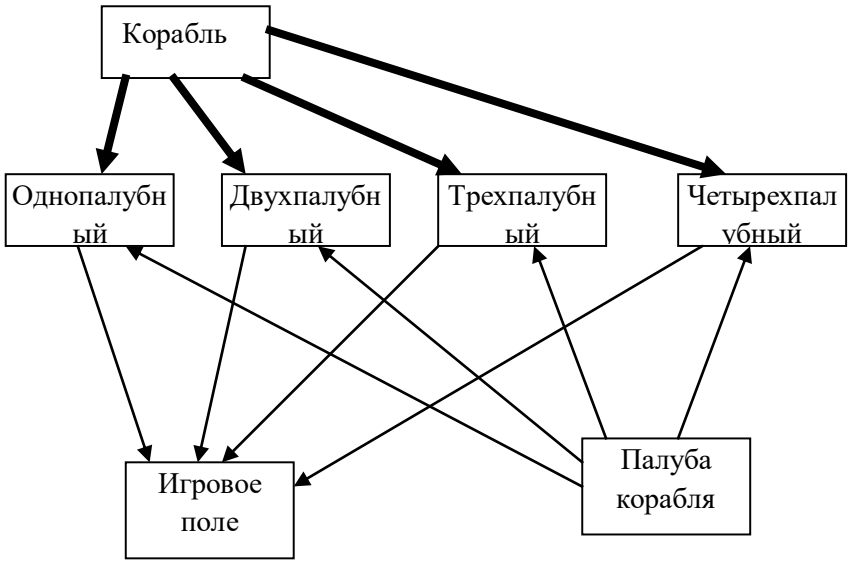
Тема 2. Игра «Питон» («Змейка»).

Использовать иерархию классов:



Класс «Фигура» должен содержать виртуальную функцию отображения. Запуск цикла игры реализовать как метод класса «Игровое поле». Сделать возможным существование на поле двух питонов, управляемых разными игроками.

Тема 3. Игра «Морской бой». Создать классы согласно следующей иерархии:



Реализовать метод отображения палубы в двух разных состояниях : целая и подбитая, а также метод отображения игрового поля (в зависимости

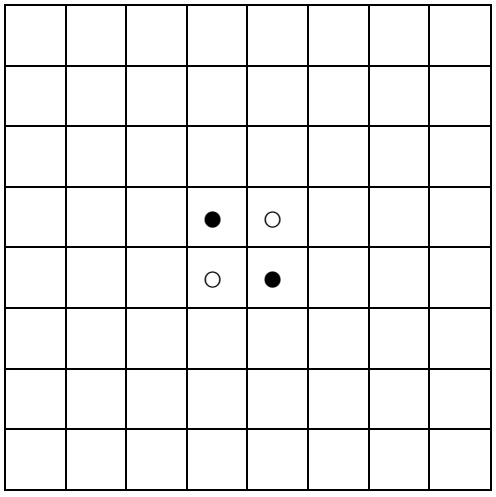
от его вида: свое поле (неподбитые корабли показаны) и чужое поле (неподбитые корабли скрыты). Реализовать для игрового поля два метода выполнения хода: с запросом (с помощью ввода с клавиатуры или управления курсором) и с помощью случайной генерации.

Тема 4. Игра «Облавные шашки» («Окружение»).

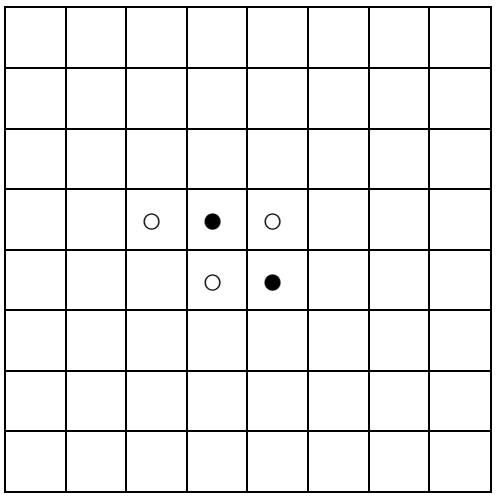
Противники поочередно делают ходы черными и белыми фишками. Когда игрок делает ход, фишки противника, оказавшиеся на прямых линиях между уже имеющимися фишками игрока, и новой фишкой, переходят к игроку (на этих прямых не должно быть пустых ячеек). Игра идет, пока не заполнены все клетки поля, или пока один из игроков не потеряет все фишки. Выигрывает тот, у кого к концу оказывается больше фишек.

а) начало игры:

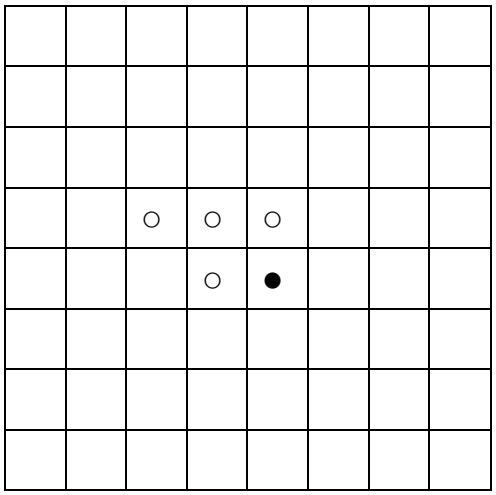
b)



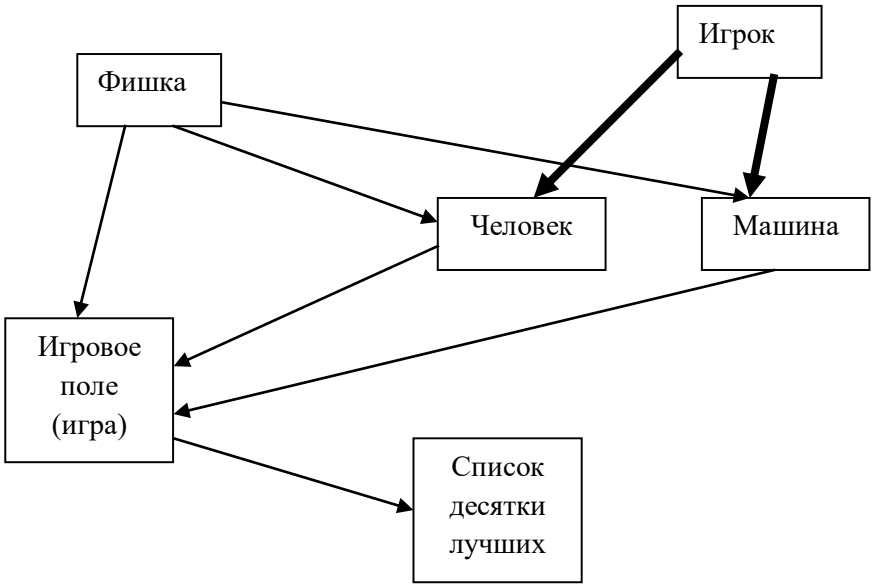
б) играющий белыми фишками сделал ход:



в) фишка противника переходит к игроку



Использовать следующую иерархию классов:



Обеспечить поддержку сохранения и загрузки игры с использованием перегрузки операций чтения/записи в поток для классов «Игровое поле» и «Игрок». Обеспечить запоминание в файле десятки лучших с помощью перегрузки этих же операторов для класса «Список десятки лучших». Класс «Игрок» должен содержать виртуальную функцию «Выполнить ход». Для человека она реализуется с запросом хода с клавиатуры. Для машины использовать «жадную» стратегию выполнения хода: при каждом ходе игрок забирает максимально возможное количество фишек противника.

Тема 5. Игра Шахматы. Реализовать класс «Игровое поле», иерархию «Фигура»-«Конкретная фигура». Реализовать метод вывода фигуры на экран. Класс «Фигура» должен содержать виртуальную функцию проверки

допустимости указанного хода, а классы для конкретных фигур – ее реализации с учетом правил выполнения ходов для каждого вида фигуры. Для класса «Игровое поле» определить методы запроса хода (поочередно для противников), выполнения хода, «съедания» фигуры противника, вывода сообщения о нахождении фигуры под боем, шахе и мате. Предусмотреть возможность сохранения и загрузки игры с помощью ввода/вывода в поток.

Тема6. Логическая игра “Крестики-нолики”

Приложение является реализацией известной логической игры “Крестики-нолики” В данной реализации предусматривается игра двух игроков на неограниченном поле. Цель игры . построить непрерывную линию из пяти или более фишек (крестиков или ноликов) по горизонтали, вертикали или диагонали. Возможны два режима игры, когда выигрывает тот, кто первым построит линию, или кто больше наберет очков за определенное количество времени. Во втором случае количество построенных линий не ограничено, каждая фишка в линии приносит игроку одно очко. Приложение должно обеспечивать начало новой игры на чистом поле, а также проверку соответствия действий игроков правилам игры и условия окончания игры. Роль одного из игроков (по выбору пользователя) может выполнять компьютер. Для исключения возможности образования изолированных игр на одном поле и неоправданного увеличения размера поля следует установить максимально допустимое расстояние (не более пяти) от уже существующих фишек до новой, размещаемой игроком.

Тема 7. Логическая игра “Цепь”.

В данной реализации предусматривается игра двух игроков на квадратном поле фиксированного размера. Цель игры - построить непрерывную линию, соединяющую горизонтальные или вертикальные (для каждого из игроков соответственно) стороны игрового поля, причем линия считается непрерывной, если фишки граничат друг с другом по горизонтали, вертикали или диагонали. Выигрывает тот игрок, который первым построит свою линию. Приложение должно обеспечивать начало новой игры на

чистом поле, а также проверку соответствия действий игроков правилам игры и условия окончания игры. Роль одного из игроков (по выбору пользователя) может выполнять компьютер.

Тема 8. Логическая игра “Точки”

В данной реализации предусматривается игра двух игроков на квадратном поле фиксированного размера. Цель игры - заполнить максимальное количество клеток игрового поля своими фишками. Игроки выполняют ходы по очереди. Ход игрока заключается в произвольной установке линии на границе двух ячеек, причем, если какая-либо ячейка оказывается со всех четырех сторон обрамлена линиями, то она отмечается фишкой данного игрока, а самому игроку предоставляется дополнительный ход, и т.д. Игра заканчивается, когда все игровое поле оказывается заполнено фишками. Выигрывает тот игрок, чьих фишек на поле больше на момент окончания игры. Приложение должно обеспечивать начало новой игры на чистом поле, а также проверку соответствия действий игроков правилам игры и условия окончания игры. Роль одного из игроков (по выбору пользователя) может выполнять компьютер.

Тема 9. Логическая игра “Мозаика”

Разрабатываемое приложение представляет собой программную реализацию популярной игры Мозаика (Puzzle), цель которой заключается в создании игроком полного изображения путем упорядочивания его фрагментов последовательным перемещением и соединением соответствующих сторон. Приложение должно обеспечивать:

просмотр исходного изображения во время игры, а также поиск подходящих друг к другу фрагментов (функция подсказки);

возможность перемещать фрагменты за пределы рабочего поля или в другое окно и обратно для улучшения наглядности.

Следует учесть, что при создании новой игры все фрагменты должны быть разъединены и размещены беспорядочно в области рабочего поля или в области, отведенной специально для фрагментов. При использовании

отдельной области для отображения фрагментов может быть применена ее прокрутка и изменение масштаба.

Тема 10. Логическая игра “Шарики”

Цель игры состоит в том чтобы набрать максимальное количество очков. Суть игры состоит в следующем: на игровом поле отображаются круги разного цвета, игроку предлагается создать линию из кругов одинакового цвета, расположенных по горизонтали или по вертикали, созданная линия .сгорает.. Причем линия может .сгореть., только в том случае если количество элементов в ней равно 3. После этой процедуры пустые места на игровом поле заполняются новыми элементами. Линию можно создать путем перестановки по горизонтали или по вертикали соседних элементов. Игра заканчивается тогда когда не возможно создать ни одной линии. Приложение должно обеспечивать возможность задания количества цветов элементов (кругов). Следует учесть что цвета кругов выбираются произвольным образом исходя из заданного количества. Кроме того, линии (3 и более элементов) получаемые при произвольной расстановки элементов “сгорают”.

Тема 11. Логическая игра “Морской бой”

Приложение должно обеспечивать:

- расстановку кораблей на игровом поле 10х10;
- выбор противника (человек, компьютер);
- изменение интерфейса в зависимости от выбора противника;
- фиксацию имен противников и число побед.

Тема 12. Логическая игра “Реверси”

Игра идет на поле произвольного размера. Два игроки по очереди устанавливают фишки своего цвета на поле. Фишку можно ставить только на те клетки, рядом с которыми уже стоят фишки. Если между установленной фишкой и какой либо другой фишкой того же цвета находятся фишки другого цвета, все они меняют свой цвет.

Тема 13. Логическая игра “Сокобан”

Играет один игрок. Игрок последовательно переходит от одного уровня к другому, по мере выполнения заданий. Уровень представляет собой лабиринт, в котором в беспорядке расставлены ящики. Игрок должен толкая ящики человечком расположить их на своих местах (места отмечены на лабиринте). Человечек может толкать только один ящик. Если ящик упирается в стену, то дальше его толкать нельзя.

Тема 13. Логическая игра “Тетрис”

Программа должна предоставлять возможность выбирать размер фигур (4, 5, 6, 7 клеток). Скорость падения управляется автоматически в зависимости от времени игры.

Тема 14. Логическая игра “Рикошет”

Игровое поле содержит источник лазера, набор целей, набор зеркал. Зеркала и цели выставляются на поле случайным образом в начале игры. Каждое зеркало может проворачиваться в любом направлении с шагом 30гр. Игрок должен поворачивая зеркала по очереди сжечь все цели.

Тема 15. Логическая игра “Авеле”

Игровое поле состоит из двух рядов, по 6 лунок в каждом. Каждая лунка в начале игры содержит 4 камня. Нижний ряд принадлежит игроку, верхний - его противнику. Игроки делают ходы по очереди. Для хода игрок берет камни из любой своей непустой лунки, и раскладывает их по одному в каждую следующую лунку, двигаясь против часовой стрелки. Начальная лунка всегда пропускается. Если игрок кладет последний камень в лунку противника И лунка содержит 2 или 3 камня (включая только что положенный) то игрок собирает камни из лунок, двигаясь по часовой стрелке, до тех пор, пока два вышеприведенных условия соблюдаются. Если игрок не может сделать очередной ход, он собирает все оставшиеся камни, игра заканчивается и определяется победитель.

Тема 17. Логическая игра “Обратный тетрис”

Тетрис наоборот. Игрок выбирает фигуру и бросает, компьютер

пытается установить ее в стакан. Цель игры - завалить компьютер.

Тема 18. Логическая игра “Шашки”

Необходимо реализовать подмножество правил классических шашек на доске 8x8.

Тема 19. Логическая игра “Уголки”

Известная игра на шашечном поле. Противники расставляют свои шашки симметрично в противоположных углах доски в виде прямоугольников 3x4. Ходы осуществляются по очереди. За один ход можно двинуть одну шашку на одну свободную клетку. Если к шашке примыкает шашка противника, а за ней есть свободное поле, можно «прыгнуть» через шашку противника на это свободное поле. За один ход можно делать несколько последовательных прыжков одной шашкой. Выигрывает тот, кто быстрее выстроит свои шашки на позиции противника.

Тема 20. Игра-аркада “Bounce”

Играют на поле, окруженном стеной, с двумя или более мячами, которые отскакивают от стен. Размер поля уменьшается, если вы создаете стену и при этом никакой шар не оказывается попавшим в ловушку. Чтобы пройти уровень, игроку за данное время нужно уменьшить размер поля по крайней мере на 75%. На каждом следующем уровне к игре добавляется по одному шару, а игроку дается больше жизней и времени. Подсчет очков зависит от того, насколько вы уменьшили площадь. Новые стену строятся по щелчку левой кнопкой мыши на свободном пространстве поля. После щелчка, начиная от клетки, где он был сделан, в противоположных направлениях начинают строиться две части стены. За один промежуток времени может строиться только одна стена.

Тема 21. Игра-аркада “Snake”

Правила игры: Чтобы выиграть в KSnake, вам нужно съесть все яблоки в комнате и выйти через дверь, которая откроется вверху. С каждым съеденным яблоком вы становитесь длиннее. Если вы врежетесь в стену, вы умрете. Если вы врежетесь в себя, вы умрете. Если вам в голову попадет мяч,

вы умрете. Если вы слишком долго не будете есть яблоки, появятся новые.

Тема 22. Игра-аркада “Космическая дуэль”

Каждый игрок управляет одним кораблем. Корабль может поворачиваться, ускоряться, стрелять и закладывать мины. У каждого корабля есть определенное количество энергии. Кораблю требуется энергия для поворотов, ускорения, стрельбы и закладывания мин. Корабль получает ее при помощи своих солнечных батарей. Количество энергии, которую корабль получает, зависит от расстояния и направления, в котором находится солнце. Корабль получает больше энергии вблизи солнца и меньше около границы. Он получает полное количество энергии, если солнце светит прямо на батареи, часть энергии, если оно светит под углом и, совсем не получает ее, если солнце светит на торец батареи. Если у корабля закончилась энергия, он теряет управление и не может стрелять. Столкновение со своими или чужими снарядами и минами уменьшает здоровье корабля. Если столкнутся два корабля, то более слабый корабль будет уничтожен, и здоровье более сильного уменьшится на значение здоровья слабого корабля, плюс некоторое значение (Ущерб при аварии). Корабль уничтожается, когда он влетает в солнце. Снаряды летают вокруг солнца как корабли. У мин есть некоторое количество энергии, чтобы оставаться на одном месте. Когда энергия заканчивается, мина падает на солнце. Минам, расположенным около солнца, требуется больше энергии, чем расположенным вдали от него. Мины могут быть уничтожены снарядами. Время от времени на поле боя появляются заправки.

Тема 23.Игра-аркада “Tron”

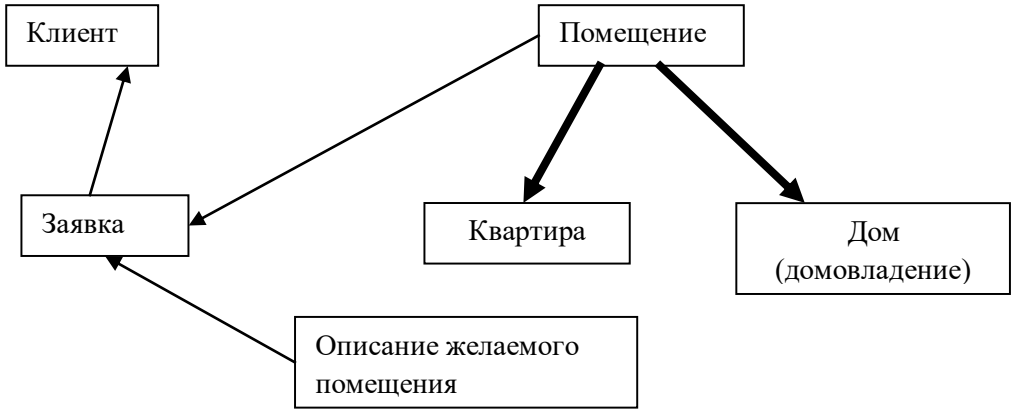
Цель игры прожить дольше, чем противник. Поэтому нельзя врезаться в стены, себя и противника. После начала раунда игроки уже не могут остановить свое движение (если не нажата пауза). Вы можете только пытаться избегать столкновения, постоянно меняя направление движения. Кроме того, вы можете мешать своему противнику. Вы можете увеличивать скорость своего движения, нажимая на клавишу ускорения. Раунд

начинается, когда все игроки нажмут на какую-нибудь клавишу направления движения. Первое движение будет совершаться в эту сторону.

КОМПЛЕКТ №4

Тема 1. Информационная система риэлтерской конторы, занимающейся продажей и обменом жилья.

Использовать следующую примерную объектную модель:

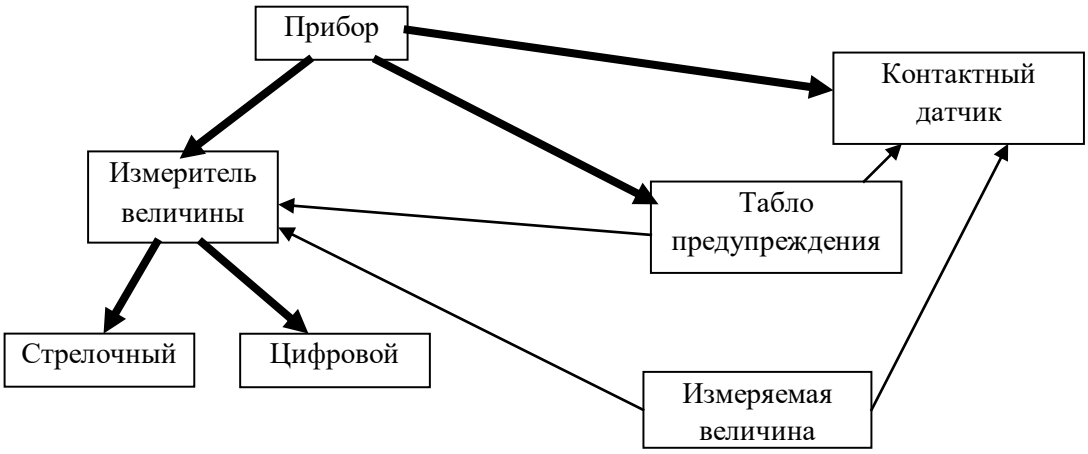


В заявке содержатся: вид операции (продажа, покупка, обмен) . Для продажи и обмена указывается имеющееся у клиента дом/квартира. Помещение описывается следующими параметрами: адрес, площадь, кол-во комнат, стоимость, наличие удобств. Для покупки и обмена к заявлению прилагается также описание желаемого помещения, где указывается, например: диапазон для цены, кол-ва комнат, требования к удобствам, требования к району размещения, и.т.п.

Реализовать функции добавления клиентов и заявок, подбора Темаов по заявкам, сохранения/загрузки объектов в файл с использованием перегрузки операторов ввода/вывода в поток.

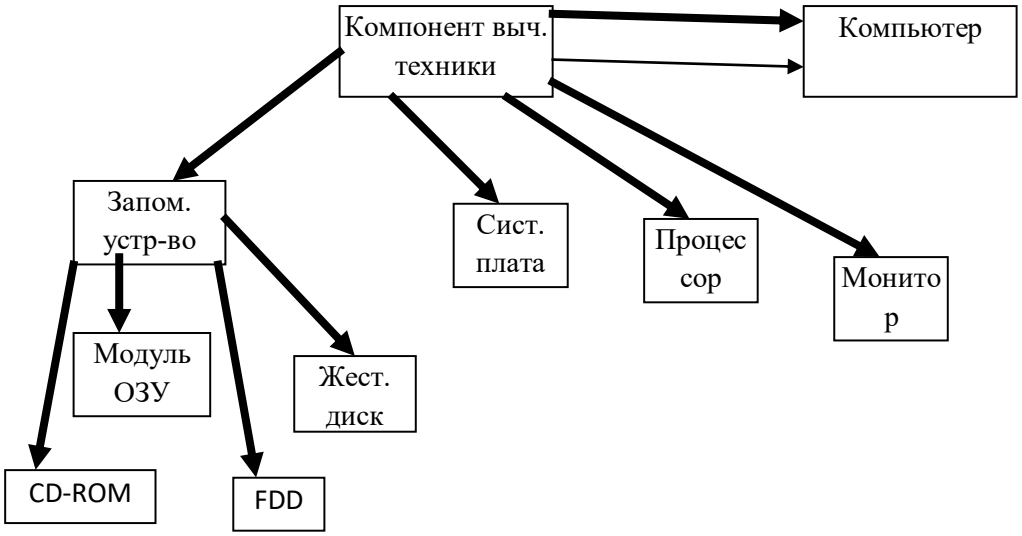
Тема 2. Виртуальная приборная панель.

Разработать обьектную модель для комплекса приборов наблюдения за какими-либо параметрами:



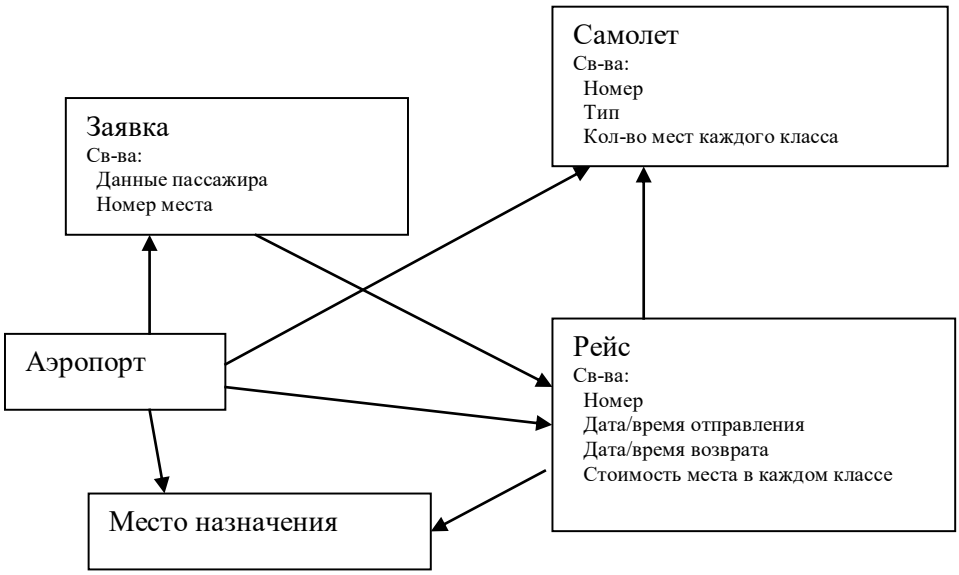
Для каждого класса приборов реализовать методы визуализации, перемещения по рабочему полю (экрану); задать набор параметров, определяющих вид прибора (напр. размер циферблата, пределы измерения, вид шрифта для цифр/текста). Реализовать методы изменения измеряемой величины. Для табло предупреждения реализовать метод, позволяющий задавать несколько диапазонов измеряемой величины и соответствующие им сообщения и вид сообщения (например: информация, предупреждение, тревога). Обеспечить для класса «Прибор» возможность периодического чтения измеряемого показателя из внешнего источника (напр. из файла). С помощью созданных классов смоделировать приборный щиток какого-либо агрегата (напр. приборная панель автомобиля, контрольные приборы газового котла и т.п.)

Тема 3. Справочная система компонентов компьютерной техники. Создать справочную систему, которая позволяла бы хранить сведения о компьютерах организации и об их составляющих (компонентах). Использовать следующую примерную иерархию классов:



Система должна позволять хранить информацию как об отдельных компонентах, так и о компонентах в составе компьютеров. Реализовать поиск компонентов по: виду, инвентарному номеру, названию производителя, признаку «свободен/находится в составе компьютера», по компонентам, принадлежащим одному компьютеру. Реализовать методы включения свободного компонента в состав компьютера; передачи компонента от одного компьютера к другому; изъятия компонента из состава компьютера и включения в состав свободных компонентов. Реализовать выгрузку в файл/ загрузку из файла всех объектов и связей между ними с помощью перегрузки операторов ввода/вывода в поток.

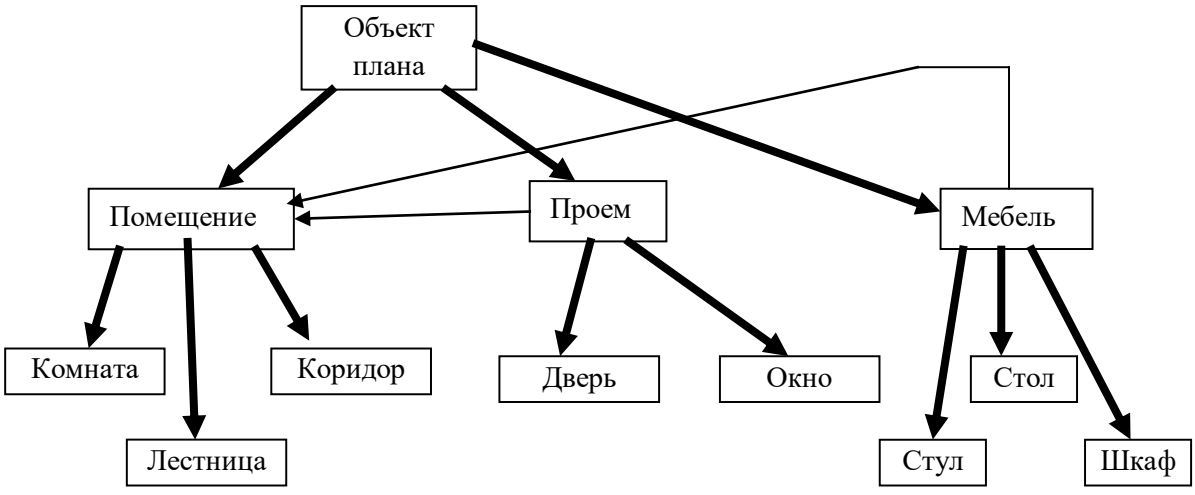
Тема 4. Информационная система «Аэропорт». Использовать приблизительно следующую иерархию:



Реализовать : назначение самолета на рейс (с контролем того, чтобы он успел вернуться из предыдущего рейса и провел некоторое время на техническое обслуживание; поиск свободного места на рейсах по месту назначения, диапазону даты и времени отправления, диапазону класса места и цены. Реализовать сохранение/загрузку состояния базы в файл с использованием перегрузки операторов ввода/вывода в поток для классов.

Тема 5. Планировщик помещений.

Разработать визуальный конструктор (редактор), позволяющий создавать планы помещений. Использовать приблизительно следующую иерархию классов для объектов, из которых может состоять план:



Реализовать методы, с помощью которых пользователь может

перемещать объекты на плане, изменять их размеры. Реализовать функцию сохранения/загрузки плана в файл с помощью перегрузки операторов ввода/вывода в поток для созданных классов.