Задача (Task) 1. Варианты задачи

1. **Рациональное число**

Разработать класс Rational (рациональное число).
Класс должен хранить корректные дроби n/m (знаменатель не равен 0).
Дробь должна храниться в несократимом виде.
Класс должен содержать все необходимые конструкторы и деструктор.
В классе должны быть перегружены операции:

- присваивания;
- 4 стандартные арифметические операции;
- сравнения;
- ввода/вывода в поток.

2. **Трехмерный вектор**

Разработать класс Vector3D, представления вектора в 3-х мерном пространстве: x, y, z.
В классе должны быть перегружены операции:

- присваивания;
- сложения и вычитания векторов;
- скалярного произведения векторов, умножение скаляра на вектор;
- сравнения векторов по длине;
- ввода/вывода в поток.

3. **Деньги**

Разработать класс Money для работы с денежными суммами, которые должны быть представлены: знаком суммы, количеством рублей и количеством копеек.
Класс должен содержать все необходимые конструкторы и деструктор.
В классе должны быть перегружены операции:

- присваивания;
- сложения и вычитания денег;
- умножения и деления денег на действительное число;
- сравнения;
- операция ввода/вывода в поток.

4. **Дата**

Разработать класс Date для работы с датами в формате, представленным в виде тройки day, month, year.
Класс должен содержать:

- все необходимые конструкторы (включая преобразования типа из строки вида “26.10:1967” и деструктор;
- метод представления даты в виде строки: “26.10:1967”;

В классе должны быть перегружены операции:

- присваивания;
- сложения (и вычитания) времени с количеством суток;
- сравнения;
- операция ввода/вывода в поток.

5. **Время**

Разработать класс Time для работы с временем в формате, представленным в виде тройки hou, min, sec.
Класс должен содержать:

- все необходимые конструкторы (включая преобразования типа из строки вида “12:24:35” и деструктор;
- метод представления времени в виде строки: “12:24:35”;

В классе должны быть перегружены операции:

- присваивания;
- сложения (и вычитания) времени с количеством секунд (переполнение результата «вверх» или «вниз» должно выполнять переход на следующие или предыдущие сутки с «отбрасываем» количества суток);
- сравнения;
- операция ввода/вывода в поток.

6. **Большое число**

Разработать класс LongLong для работы с целыми 64 битовыми числами, представленными двумя полями: int — старшая часть, unsigned int — младшая часть каждые длиной по 32 бита.
Класс должен содержать все необходимые конструкторы и деструктор.
В классе должны быть перегружены операции:

- присваивания;
- 4 стандартные арифметические операции;
- сравнения;
- ввода/вывода в поток.

Задача (Task) 2

1. Вектор

Разработать класс **Vector**.
Класс должен хранить вектор (массив) целых чисел заданного размера, размещенный в динамической памяти.
Класс должен содержать все необходимые конструкторы, деструктор.
Класс должен предоставлять следующие операции и методы:
1) операции присваивания, суммирования и скалярного произведения векторов
2) операцию индексации с контролем выхода индекса за границы массива.
3) операцию умножения скаляра на вектор
4) операции << и >> для сохранения себя в файле и чтения из файла
5) вычислить длину вектора.

2. Полином

Разработать класс **Polinom**. Класс должен хранить полином (многочлен) от одной переменной (x), представленный в виде степени полинома и массива действительных коэффициентов полинома, размещенного в динамической памяти.
Класс должен содержать все необходимые конструкторы, деструктор.
Класс должен предоставлять следующие операции и методы:
1) операции присваивания и суммирования
2) операцию индексации с контролем выхода индекса за границы массива.
3) метод вывода себя на консоль в текстовой форме: 1.5+4.8*x+2.4*x^2+ …
4) операции << и >> для сохранения себя в файле и чтения из файла
5) вычислить значение полинома в заданной точке x,
6) найти полином, являющийся производной исходного полинома.

3. Большое число

Разработать класс Decimal для работы с беззнаковыми целыми десятичными числами, используя для представления числа массив из элементов типа unsigned char, каждый из которых является десятичной цифрой. Младшая цифра имеет меньший индекс (единицы — в нулевом элементе массива). Размер массива задается при конструировании объектов.
Класс должен содержать все необходимые конструкторы и деструктор.
В классе должны быть перегружены операции:

- присваивания;
- сложения и вычитания;
- сравнения;
- ввода/вывода в поток.

4. Восьмеричное число

Создать класс **Octal** для работы с беззнаковыми целыми восьмеричными числами, используя для представления числа массив элементов типа unsigned char, каждый элемент которого является восьмеричной цифрой. Младшая цифра имеет меньший индекс (единицы — в нулевом элементе массива). Размер массива задается как аргумент конструктора.
Реализовать операции:

присваивания;
арифметические: +, -
сравнения
можно попробовать умножение
операцию индексации с контролем выхода индекса за границы массива.
операции << и >> для сохранения себя в файле и чтения из файла

5. Шестнадцатеричное число

Создать класс **Hex** для работы с беззнаковыми целыми шестнадцатеричными числами, используя для представления числа массив из элементов типа unsigned char, каждый из которых является шестнадцатеричной цифрой. Младшая цифра имеет меньший индекс. Размер массива задается как аргумент конструктора.
Реализовать операции:
присваивания;
арифметические: +, -;
сравнения
можно попробовать умножение
операцию индексации с контролем выхода индекса за границы массива.
операции << и >> для сохранения себя в файле и чтения из файла

6. Матрица

Разработать класс **Matrix**.
Класс должен хранить квадратную матрицу целых чисел заданного размера.
Класс должен содержать все необходимые конструкторы, деструктор
Класс должен предоставлять следующие операции и методы:
1) операцию присваивания
2) операцию суммирования двух матриц одинакового размера
3) операцию индексации с контролем выхода индекса за границы массива.De
4) операции << и >> для сохранения себя в файле и чтения из файла
5) операцию транспонирования матрицы
6) проверить, обладает ли матрица диагональным преобладанием,
7) операцию умножения матриц и операцию умножения скаляра на матрицу

Задача (Task) 3

1. Табулятор функции

Разработать класс **Tabulator**.
Класс должен позволять выполнять табулирование произвольной функции одной переменной, заданной в виде функции f(x) языка C++.
Табулирование функции f(x) на заданном интервале табулирования [a, b] и заданном количестве точек табуляции n состоит в сформировании динамических массивов значений X в равноотстоящих точках табуляции и значений функции F в этих точках в виде:

$X: \quad x_0=a, \quad x_1=a+h, \quad x_2=a+2h, \ . \ . \ . \ x_{n-1}=b$

$F: \quad f_0=f(x_0), \ f_1=f(x_1), \ f_2=f(x_2), \ . \ . \ . \ f_{n-1}=f(x_{n-1}),$

Где h = (b – a) / (n-1)
Вычисление табулированной функции в точке x (a <= x <=b) выполняется с помощью линейной интерполяции по двум точкам плоскости: (x_i, f_i) и (x_{i+1}, f_{i+1}), где: $x_i <= x <= x_{i+1}$.

Класс должен содержать необходимые конструкторы и деструктор.
Класс должен предоставлять следующие операции (методы класса):
1) задать интервал табулирования a, b;
2) задать число точек табулирования n,
3) узнать интервал табулирования,
4) узнать число точек табулирования,
5) выполнить табулирование функции,
6) вычислить значение табулированной функции в заданной точке x и сравнить это значение со значением исходной функции в этой точке.

2. Расчет интегралов

Разработать класс **Integrals**.
Класс должен позволять вычислять приближенное значение интеграла от произвольной функции одной переменной, заданной в виде функции языка C++.
Класс должен для заданного интервала интегрирования [a, b] и количества отрезков разбиения интервала n вычислять приближенное значение интеграла, используя методы левых, правых и средних прямоугольников.
Класс должен содержать необходимые конструкторы и деструктор.
Класс должен предоставлять следующие операции (методы класса):
1) конструктор инициализатор с параметрами: интервал интегрирования, число отрезков разбиения интервала интегрирования;
4) задать интегрируемую функцию;
6) вычислить значение интеграла указанным методом;
7) вывести результат вычисления на экран.

3. Ряд Тейлора

Разработать класс **Taylor**.
Ряд Тейлора – приближенное представление значения функции f(x) в окрестности точки a в виде суммы в виде конечной суммы членов ряда Тейлора:

f(x) = $f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \dots + \frac{f^{(n)}(a)}{n!}(x-a)^n + \dots$

Класс должен формировать динамический массив значений коэффициентов ряда (типа double) Тейлора: f(a), f'(a)/1!, f''(a)/2!, … для заданного числа членов ряда. В качестве функции на усмотрение студента может быть выбрана одна из: sin(x), cos(x), exp(x).

Класс должен содержать необходимые конструкторы и деструктор.

Класс должен предоставлять следующие операции (методы класса):

1) задать число членов ряда,
2) узнать текущее число членов ряда,
3) задать значение параметра a и подсчитать значения массива коэффициентов
4) выдать формулу ряда с коэффициентами, подсчитанными для заданного параметра a,
5) выдать значение заданного члена ряда,
6) рассчитать значение ряда в выбранной точке x,
7) вывести отклонение значения ряда в выбранной точке от эталонного значения функции в данной точке (эталонное значение вычисляется, используя соответствующую функцию из стандартной библиотеки C++).
8) перегрузить операцию записи – чтения из потока (файлового)

4. Пользовательское меню

Разработать класс **Menu**

Класс должен предоставлять одноуровневое (однострочное) меню с заданным числом команд в консольном режиме экрана.

Класс должен содержать необходимые конструкторы, деструктор и методы (по выбору студента).

В качестве обязательных, класс должен содержать методы:

1) Конструктор инициализатор с параметрами:
    a. Местоположение меню: координаты начала строки меню и длину меню.
    b. Количество пунктов меню
    c. Массив названий пунктов меню
2) Конструктор с параметром – имя текстового файла описания меню. Текст описания меню должен содержать:
    a. Местоположение меню в виде: xCor yCor sise
    b. Количество пунктов меню
    c. Названия пунктов (по одному в строке)

Пример:

1 1 80
3
File
Main
View

В этом примере указано, что меню:

- начинается в позиции экрана с координатами (1, 1) и имеет длину 80 символов;
- включает 3 пункта;
- которые имеют имена: File, Main и View.

3) Вывести (показать) меню на экран,
4) Удалить меню с экрана,
5) Выбрать пункт меню с помощью клавиш: стрелки (влево, вправо) и ввод (Enter) с выдачей номера выбранного пункта меню.

Работа с консолью - См. http://www.c-cpp.ru/funkcii/conioh

5. Однострочный текстовый редактор

Разработать класс **TextEditor**.
Класс должен предоставлять возможность разместить в выбранной позиции окна консоли поле заданной длины для ввода с клавиатуры последовательности символов. В минимальном варианте длина последовательности не должна превышать длину поля ввода.
Класс должен содержать необходимые служебные методы (конструкторы, деструктор и пр.).
Класс должен предоставлять следующие операции:

1) конструктор инициализатор с параметрами: начальная позиция поля ввода (x, y) в окне консоли. длина поля ввода;
2) показать текстовый редактор, убрать с экрана текстовый редактор;
3) обеспечить ввод пользователем строки с длиной не больше длины поля ввода;
4) выдать введенную пользователем строку.

Программа должна иметь простейшее меню:
- Введите положение и размер окна
- Показать редактор
- Убрать редактор
- Введите и отредактируйте текст
- Завершить работу

Работа с консолью - См. http://www.c-cpp.ru/funkcii/conioh

6. Словарь переводчика

Разработать класс **Dictionary**.
Класс должен предоставлять возможность формировать англо-русский словарь. В минимальном варианте каждому английскому слову соответствует ровно одно русское слово-перевод.
Класс должен содержать необходимые служебные методы (конструкторы, деструктор и пр.).
Класс должен предоставлять следующие операции:
1) добавить в словарь слово и его перевод,
2) изменить перевод указанного слова,
3) узнать перевод выбранного слова,
4) проверить наличие слова в словаре,
5) узнать число слов в словаре,
6) сохранить словарь в файл, считать словарь из файла.
7) перегрузить операции присваивания и объединения словарей.

Задача (Task) 3_1

1. Расчет интегралов

Разработать класс **Integrals**.
Класс должен позволять вычислять приближенное значение интеграла от произвольной функции одной переменной, заданной в виде функции языка C++.
Класс должен для заданного интервала интегрирования [a, b] и количества отрезков разбиения интервала n вычислять приближенное значение интеграла, используя методы левых, правых и средних прямоугольников.
Класс должен содержать необходимые конструкторы и деструктор.
Класс должен предоставлять следующие операции (методы класса):
1) конструктор инициализатор с параметрами: интервал интегрирования, число отрезков разбиения интервала интегрирования;
2) задать интегрируемую функцию;
3) вычислить значение интеграла указанным методом;
4) вывести результат вычисления на экран.

2. Табулятор функции

Разработать класс **Tabulator**.
Класс должен позволять выполнять табулирование произвольной функции одной переменной, заданной в виде функции f(x) языка C++.
Табулирование функции f(x) на заданном интервале табулирования [a, b] и заданном количестве точек табуляции n состоит в сформировании динамических массивов значений X в равноотстоящих точках табуляции и значений функции F в этих точках в виде:

X: $x_0=a$, $x_1=a+h$, $x_2=a+2h$, . . . $x_{n-1}=b$
F: $f_0=f(x_0)$, $f_1=f(x_1)$, $f_2=f(x_2)$,. . . $f_{n-1}=f(x_{n-1})$,

Где h = (b – a) / (n-1)
Для вычисления значений табулированной функции использовать в качестве тестовой одну из элементарных функций: sin, cos, exp
Вычисление табулированной функции в точке x (a <= x <=b) выполняется с помощью линейной интерполяции по двум точкам плоскости: (x_i, f_i) и (x_{i+1}, f_{i+1}), где: x_i <= x <= x_{i+1}.

Класс должен содержать необходимые конструкторы и деструктор.
Класс должен предоставлять следующие операции (методы класса):
1) выполнить табулирование для заданной тестовой функции,
2) вычислить значение табулированной функции в заданной точке x и сравнить это значение со значением тестовой функции в этой точке.
3) перегрузить операцию записи – чтения из потока (файлового)

3. Ряд Тейлора

Разработать класс **Taylor**.
Ряд Тейлора – приближенное представление значения функции f(x) в окрестности точки a в виде суммы в виде конечной суммы членов ряда Тейлора:

f(x) = $f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \ldots + \frac{f^{(n)}(a)}{n!}(x-a)^n + \ldots$

Класс должен формировать динамический массив значений коэффициентов ряда (типа double) Тейлора: f(a), f'(a)/1!, f''(a)/2!, … для заданного числа членов ряда. В качестве функции на усмотрение студента может быть выбрана одна из: sin(x), cos(x), exp(x).
Класс должен содержать необходимые конструкторы и деструктор.
Класс должен предоставлять следующие операции (методы класса):
1) задать значение параметра a и подсчитать значения массива коэффициентов
2) выдать формулу ряда с коэффициентами, подсчитанными для заданного параметра a,
3) выдать значение заданного члена ряда,
4) рассчитать значение ряда в выбранной точке x,
5) вывести отклонение значения ряда в выбранной точке от эталонного значения функции в данной точке (эталонное значение вычисляется, используя соответствующую функцию из стандартной библиотеки C++).
6) перегрузить операцию записи – чтения из потока (файлового)

4. Пользовательское меню

Разработать класс **Menu**
Класс должен предоставлять одноуровневое (однострочное) меню с заданным числом команд в консольном режиме экрана.
Класс должен содержать необходимые конструкторы, деструктор и методы (по выбору студента).
В качестве обязательных, класс должен содержать методы:

6) Конструктор инициализатор с параметрами:
    a. Местоположение меню: координаты начала строки меню и длину меню.
    b. Количество пунктов меню
    c. Массив названий пунктов меню
7) Конструктор с параметром – имя текстового файла описания меню. Текст описания меню должен содержать:
    a. Местоположение меню в виде: xCor yCor sise
    b. Количество пунктов меню
    c. Названия пунктов (по одному в строке)

Пример:
1 1 80
3
File
Main
View
В этом примере указано, что меню:
- начинается в позиции экрана с координатами (1, 1) и имеет длину 80 символов;
- включает 3 пункта;
- которые имеют имена: File, Main и View.

8) Вывести (показать) меню на экран,
9) Удалить меню с экрана,
10) Выбрать пункт меню с помощью клавиш: стрелки (влево, вправо) и ввод (Enter) с выдачей номера выбранного пункта меню.

Работа с консолью - См. http://www.c-cpp.ru/funkcii/conioh

5. Однострочный текстовый редактор

Разработать класс **TextEditor**.

Класс должен предоставлять возможность разместить в выбранной позиции окна консоли поле заданной длины для ввода с клавиатуры последовательности символов. В минимальном варианте длина последовательности не должна превышать длину поля ввода.
Класс должен содержать необходимые служебные методы (конструкторы, деструктор и пр.).
Класс должен предоставлять следующие операции:

5) конструктор инициализатор с параметрами: начальная позиция поля ввода (x, y) в окне консоли. длина поля ввода;
6) показать текстовый редактор, убрать с экрана текстовый редактор;
7) обеспечить ввод пользователем строки с длиной не больше длины поля ввода;
8) выдать введенную пользователем строку.

Программа должна иметь простейшее меню:
- Введите положение и размер окна
- Показать редактор
- Убрать редактор
- Введите и отредактируйте текст
- Завершить работу

Работа с консолью - См. http://www.c-cpp.ru/funkcii/conioh

6. Словарь переводчика

Разработать класс **Dictionary**.
Класс должен предоставлять возможность формировать англо-русский словарь. В этом словаре каждому английскому слову соответствует ровно одно русское слово-перевод.
Класс должен содержать необходимые служебные методы (конструкторы, деструктор и пр.).
Класс должен предоставлять следующие операции:
1) добавить в словарь слово и его перевод,
2) изменить перевод указанного слова,
3) узнать перевод выбранного слова,
4) проверить наличие слова в словаре,
5) узнать число слов в словаре,
6) перегрузить операции записи, чтения словаря в поток.
7) перегрузить операции присваивания и объединения словарей.

Задача (Task) 4. Варианты задачи

1. Термометр

Разработать класс **Thermometer**.
Класс должен хранить историю наблюдений за температурой в течение одного календарного (не високосного) года. Наблюдения описываются датой (день, месяц, год) и временем (в часах). При поступлении нового наблюдения для уже существующих даты и времени старое наблюдение заменяется. Температура задается в градусах Цельсия.
Начальная дата и время наблюдений устанавливается при конструировании объектов и в процессе работы не меняется.
Класс должен содержать необходимые служебные методы (конструкторы, деструктор, …).
Класс должен предоставлять следующие операции:
8) сохранить историю наблюдений в файл и считать историю наблюдений из файла.
1) узнать начальные дату и время наблюдений,
2) добавить наблюдение,
3) узнать температуру в наблюдении, выбранном по дате и времени,
4) выдать серию наблюдений для выбранной даты,
5) найти среднюю температуру для выбранной даты, выбранного месяца, за всю историю наблюдений,
7) найти среднюю дневную или ночную температуру для выбранного месяца,
Программа должна иметь простое меню для выбора необходимых операций.

2. Весы напольные

Разработать класс **FloorScales**.
Класс должен хранить историю наблюдений за показаниями веса членов семьи (до пяти человек). Показания описываются датой (день, месяц, год) и именем члена семьи. При поступлении нового наблюдения для уже существующей даты старое наблюдение заменяется. Вес задается в килограммах с точностью до 50 граммов.
Начальная дата и время наблюдений устанавливается при конструировании объектов и в процессе работы не меняется.
Класс должен содержать необходимые служебные методы (конструкторы, деструктор, …).
Класс должен предоставлять следующие операции:
7) сохранить историю наблюдений в файл и считать историю наблюдений из файла.
1) узнать начальные дату и время наблюдений,
2) задать наблюдение,
3) узнать вес в наблюдении, выбранном по дате и имени члена семьи,
4) найти средний вес члена семьи в выбранном месяце или за всю историю наблюдений,
5) найти минимальный вес члена семьи в выбранном месяце или за всю историю наблюдений и дату, когда он наблюдался
6) найти максимальный вес члена семьи в выбранном месяце или за всю историю наблюдений и дату, когда он наблюдался,
Программа должна иметь простое меню для выбора необходимых операций.

3. Шагомер

Разработать класс **Pedometer**.
Класс должен хранить историю подсчета шагов владельца. Каждый подсчет описывается датой (день, месяц, год) и интервалом времени (час, минута начала движения, час, минута окончания движения). Подсчет ведется в шагах с точностью до единицы.
Класс должен содержать необходимые служебные методы (конструкторы, деструктор, …)..
Начальная дата и время наблюдений устанавливается при конструировании объектов и в процессе работы не меняется.
Класс должен предоставлять следующие операции:
6) сохранить историю подсчетов в файл и считать историю подсчетов из файла.
1) узнать начальные дату и время подсчетов,
2) добавить подсчет,
3) получить информацию о подсчете, выбранном по дате и интервалу времени,
4) найти среднее число шагов в выбранном месяце или за всю историю наблюдений,
5) найти максимальное число шагов в день в выбранном месяце или за всю историю наблюдений и дату, когда оно было достигнуто,
Программа должна иметь простое меню для выбора необходимых операций.

4. Контакты

Разработать класс **Contacts**.
Класс должен хранить информацию о контактах владельца. Каждый контакт содержит следующие данные: фамилия; имя; отчество; телефон; день рождения (день, месяц, год); признак, относится ли контакт к избранным. Контакты хранятся упорядоченно по фамилии, имени, отчеству. Фамилия, имя, отчество (ФИО) являются обязательными полями. Данные вводятся на русском языке.
Класс должен содержать необходимые служебные методы (конструкторы, деструктор, …)..
Класс должен предоставлять следующие операции: ,
11) сохранить контакты в файл и считать контакты из файла.
1) создать новый контакт, 2) найти контакт по ФИО; 3) изменить контакт, выбранный по ФИО, 4) найти контакт по телефону, 5) выдать все контакты на заданную начальную букву ФИО, 6) узнать текущее число контактов, 7) внести контакт, выбранный по ФИО в список избранных, 8) удалить контакт, выбранный по ФИО из списка избранных, 9) выдать все избранные контакты, 10) удалить контакт, выбранный по телефону
Программа должна иметь простое меню для выбора необходимых операций.

5. Песенник

Разработать класс **Songs**.
Класс должен хранить информацию о песенных композициях. Каждая песня описывается следующими данными: название, поэт (автор стихов), композитор (автор музыки), исполнитель, название альбома (если входит в какой-то альбом), дата выпуска (день, месяц, год). Песни хранятся упорядоченно по названию. Данные вводятся на русском языке.

Класс должен содержать необходимые служебные методы (конструкторы, деструктор, …)..
Класс должен предоставлять следующие операции:
9) сохранить песенник в файл и считать песенник из файла.
1) добавить песню, 2) изменить данные песни, выбранной по названию, 3) найти песню по названию и исполнителю, 4) выдать все песни заданного поэта, 5) выдать все песни заданного композитора, 6) выдать все песни заданного исполнителя, 7) узнать текущее число песен, 8) удалить песню по названию,
Программа должна иметь простое меню для выбора необходимых операций.

6. Фильмотека

Разработать класс **FilmLibrary**.
Класс должен хранить информацию о фильмах. Каждый фильм описывается следующими данными: название, режиссер, сценарист, композитор, дата выхода в прокат (день, месяц, год), сборы (в рублях). Фильмы хранятся упорядоченно по названию и годам. Данные вводятся на русском языке.
Класс должен содержать необходимые служебные методы (конструкторы, деструктор, …)..
Класс должен предоставлять следующие операции:
10) сохранить фильмотеку в файл и считать фильмотеку из файла.
1) добавить фильм, 2) изменить данные фильма выбранного по названию, 3) найти фильм по названию и году, 4) выдать все фильмы заданного режиссера, 5) выдать все фильмы, вышедшие в прокат в выбранном году, 6) выдать заданное число фильмов с наибольшими сборами, 7) выдать заданное число фильмов с наибольшими сборами в выбранном году, 8) узнать текущее число фильмов, 9) удалить фильм по названию,
Программа должна иметь простое меню для выбора необходимых операций.

Задача (Task) 4_1. Варианты задачи

1. Термометр

Разработать класс **Thermometer**.
Класс должен хранить историю наблюдений за температурой в течение одного календарного (не високосного) года. Наблюдения описываются датой (день, месяц, год) и временем (в часах). При поступлении нового наблюдения для уже существующих даты и времени старое наблюдение заменяется. Температура задается в градусах Цельсия.
Начальная дата и время наблюдений устанавливается при конструировании объектов и в процессе работы не меняется.
Класс должен содержать необходимые служебные методы (конструкторы, деструктор, …).
Класс должен предоставлять следующие операции:
8) сохранить историю наблюдений в файл и считать историю наблюдений из файла.
1) узнать начальные дату и время наблюдений,
2) добавить наблюдение,
3) узнать температуру в наблюдении, выбранном по дате и времени,
4) выдать серию наблюдений для выбранной даты (не на консоль),
5) найти среднюю температуру для выбранной даты, выбранного месяца, за всю историю наблюдений,
7) найти среднюю дневную или ночную температуру для выбранного месяца,
Программа должна иметь простое меню для выбора необходимых операций.

2. Весы напольные

Разработать класс **FloorScales**.
Класс должен хранить историю наблюдений за показаниями веса членов семьи (до пяти человек). Показания описываются датой (день, месяц, год) и именем члена семьи. При поступлении нового наблюдения для уже существующей даты старое наблюдение заменяется. Вес задается в килограммах с точностью до 50 граммов.
Начальная дата и время наблюдений устанавливается при конструировании объектов и в процессе работы не меняется.
Класс должен содержать необходимые служебные методы (конструкторы, деструктор, …).
Класс должен предоставлять следующие операции:
7) сохранить историю наблюдений в файл и считать историю наблюдений из файла.
1) узнать начальные дату и время наблюдений,
2) задать наблюдение,
3) узнать вес в наблюдении, выбранном по дате и имени члена семьи,
4) найти средний вес члена семьи в выбранном месяце или за всю историю наблюдений,
5) найти минимальный вес члена семьи в выбранном месяце или за всю историю наблюдений и дату, когда он наблюдался
6) найти максимальный вес члена семьи в выбранном месяце или за всю историю наблюдений и дату, когда он наблюдался,
Программа должна иметь простое меню для выбора необходимых операций.

3. Шагомер

Разработать класс **Pedometer**.
Класс должен хранить историю подсчета шагов владельца. Каждый подсчет описывается датой (день, месяц, год) и интервалом времени (час, минута начала движения, час, минута окончания движения). Подсчет ведется в шагах с точностью до единицы.
Класс должен содержать необходимые служебные методы (конструкторы, деструктор, …)..
Начальная дата и время наблюдений устанавливается при конструировании объектов и в процессе работы не меняется.
Класс должен предоставлять следующие операции:
6) сохранить историю подсчетов в файл и считать историю подсчетов из файла.
1) узнать начальные дату и время подсчетов,
2) добавить подсчет,
3) получить информацию о подсчете, выбранном по дате и интервалу времени,
4) найти среднее число шагов в выбранном месяце или за всю историю наблюдений,
5) найти максимальное число шагов в день в выбранном месяце или за всю историю наблюдений и дату, когда оно было достигнуто,
Программа должна иметь простое меню для выбора необходимых операций.

4. Контакты

Разработать класс **Contacts**.
Класс должен хранить информацию о контактах владельца. Каждый контакт содержит следующие данные: фамилия; имя; отчество; телефон; день рождения (день, месяц, год); признак, относится ли контакт к избранным. Контакты хранятся упорядоченно по фамилии, имени, отчеству. Фамилия, имя, отчество (ФИО) являются обязательными полями. Данные вводятся на русском языке.
Класс должен содержать необходимые служебные методы (конструкторы, деструктор, …)..
Класс должен предоставлять следующие операции: ,
11) сохранить контакты в файл и считать контакты из файла.
1) создать новый контакт, 2) найти контакт по ФИО; 3) изменить контакт, выбранный по ФИО, 4) найти контакт по телефону, 5) выдать все контакты на заданную начальную букву ФИО (не на консоль), 6) узнать текущее число контактов, 7) внести контакт, выбранный по ФИО в список избранных, 8) удалить контакт, выбранный по ФИО из списка избранных, 9) выдать все избранные контакты (не на консоль), 10) удалить контакт, выбранный по телефону
Программа должна иметь простое меню для выбора необходимых операций.

5. Песенник

Разработать класс **Songs**.
Класс должен хранить информацию о песенных композициях. Каждая песня описывается следующими данными: название, поэт (автор стихов), композитор (автор музыки), исполнитель, название альбома (если входит в какой-то альбом), дата выпуска (день, месяц, год). Песни хранятся упорядоченно по названию. Данные вводятся на русском языке.

Класс должен содержать необходимые служебные методы (конструкторы, деструктор, …)..
Класс должен предоставлять следующие операции:
9) сохранить песенник в файл и считать песенник из файла.
1) добавить песню, 2) изменить данные песни, выбранной по названию, 3) найти песню по названию и исполнителю, 4) выдать все песни заданного поэта (не на консоль),, 5) выдать все песни заданного композитора (не на консоль),, 6) выдать все песни заданного исполнителя (не на консоль),, 7) узнать текущее число песен, 8) удалить песню по названию,
Программа должна иметь простое меню для выбора необходимых операций.

6. Фильмотека

Разработать класс **FilmLibrary**.
Класс должен хранить информацию о фильмах. Каждый фильм описывается следующими данными: название, режиссер, сценарист, композитор, дата выхода в прокат (день, месяц, год), сборы (в рублях). Фильмы хранятся упорядоченно по названию и годам. Данные вводятся на русском языке.
Класс должен содержать необходимые служебные методы (конструкторы, деструктор, …)..
Класс должен предоставлять следующие операции:
10) сохранить фильмотеку в файл и считать фильмотеку из файла.
1) добавить фильм, 2) изменить данные фильма выбранного по названию, 3) найти фильм по названию и году, 4) выдать все фильмы заданного режиссера (не на консоль),, 5) выдать все фильмы, вышедшие в прокат в выбранном году (не на консоль),, 6) выдать заданное число фильмов с наибольшими сборами (не на консоль),, 7) выдать заданное число фильмов с наибольшими сборами в выбранном году (не на консоль),, 8) узнать текущее число фильмов, 9) удалить фильм по названию,
Программа должна иметь простое меню для выбора необходимых операций.

Задача (Task) 5. Варианты задачи

5.1. Касса и Склад

Разработать классы **Kassa** и **Sklad**.
Класс **Касса** должен имитировать работу кассового аппарата по сканированию товаров и формированию чека за покупку. Каждый товар идентифицируется штрих-кодом (для упрощения – строка из четырех цифр от «0001» до «9999»). Один и тот же товар может сканироваться несколько раз, но в чек информация о каждом товаре входит в виде «наименование – стоимость за единицу (для упрощения в рублях без копеек) – количество – общая стоимость за товар». Чек состоит не менее чем из одной записи указанного вида. Чек дополнительно включает общую стоимость товаров в покупке, суммарную скидку и итоговую сумму к оплате (все в рублях).
База товаров хранится в классе **Склад**. Товар описывается штрих-кодом, наименованием, стоимостью за единицу товара, скидкой в процентах от стоимости. Скидки устанавливаются на каждый товар независимо (в диапазоне от 1 до 50%).
Класс **Касса** должен предоставлять следующие операции: 1) «сканировать» очередной товар, 2) получить описание товара со склада, 3) добавить данные о товаре в чек, 4) сформировать чек за покупку, 5) рассчитать итоговую сумму к оплате, 6) удалить выбранный товар из покупки.
Класс **Склад** должен использоваться для поддержки работы класса **Касса** и может быть разработан в минимально-необходимом варианте.

5.2. Банкомат и Процессинговый центр

Разработать классы **Bankomat** и **ProcCenter**.
Класс **Bankomat** должен имитировать работу банкомата по приему и выдаче наличных денежных средств (в рублях).
Класс должен поддерживать работу с купюрами в 100, 200, 500, 1000, 2000, 5000 рублей. Купюры каждого достоинства хранятся в отдельной кассете. Емкость каждой кассеты – 2000 купюр. Считать, что начальная загрузка банкомата составляет 80% для каждой кассеты.
Выдаваемая или принимаемая за одну операцию сумма ограничена 40 купюрами (независимо от их достоинства). Считать, что клиент банка идентифицируется по номеру пластиковой карты (для упрощения – номер карты от «0001» до «9999»).
База клиентов хранится в классе **ProcCenter**. Номера выданных клиентам карт могут идти не подряд. Считать, что информация о клиенте состоит из номера карты, ФИО владельца, суммы на счету, (для упрощения – без копеек), PIN-кода (последовательность из 4-х цифр, каждая от 0 до 9).
Класс **Bankomat** должен предоставлять следующие операции: 1) принять карту клиента, 2) найти клиента по номеру карты, 3) проверить PIN-код, 4) распечатать состояние счета клиента, 5) выдать клиенту наличные (списав выданную сумму со счета), 6) принять от клиента наличные (зачислив полученную сумму на счет), 7) заблокировать карту клиента, если до ее возврата клиенту три раза подряд набран неверный PIN-код, 8) вернуть карту клиенту.
Все операции должны сопровождаться необходимыми проверками на указанные выше ограничения.
Класс **ProcCenter** должен использоваться для поддержки работы класса **Bankomat** и может быть разработан в минимально-необходимом варианте.

5.3. Депозит и Процессинговый центр

Разработать классы **Depozit** и **ProcCenter**.
Класс **Depozit** должен имитировать работу интернет-банка в части управления депозитом для зарплатных клиентов банка (тех, чья зарплата перечисляется работодателем на счета работников, открытые в данном банке). Депозит может открываться клиентом на выбранный срок. Возможные варианты: 3 месяца, 6 месяцев, 1 год, 2 года, 3 года. Считать, что процентные ставки по депозитам зависят только от срока депозита и начальной суммы. Диапазоны начальных сумм: до 100 тыс. рублей, от 100 до 500 тыс. рублей, от 500 тыс. до 1 млн рублей, свыше 1 млн рублей. Процентные ставки указываются для срока 1 год (например, 6% годовых по вкладу на 6 месяцев от 100 до 500 тыс. рублей или 6.6% годовых по вкладу на 1 год от 500 тыс. до 1 млн рублей). Считать, что проценты начисляются раз в месяц на начальную сумму депозита и могут быть сняты клиентом в любой момент. Депозит может быть закрыт только по истечении срока, на который он был открыт.
База клиентов хранится в классе **ProcCenter**. Считать, что информация о клиенте состоит из номера зарплатного счета (для упрощения – номер счета от «0001» до «9999»), ФИО владельца, суммы на зарплатном счету, (для упрощения – без копеек), информации о депозите, пароля (произвольная строка, выбранная пользователем, с длиной больше 3 символов). Считать, что авторизация клиента в интернет-банке происходит по номеру зарплатного счета и паролю.
Класс **Depozit** должен предоставлять следующие операции: 1) авторизовать клиента, 2) показать информацию о доступных клиенту депозитах, исходя из суммы на его зарплатном счету, 3) проверить наличие открытого депозита, 4) открыть депозит (переведя указанную сумму с зарплатного счета клиента на депозит), 5) показать состояние депозита, 6) снять проценты (переведя их на зарплатный счет клиента), 7) закрыть депозит (переведя всю накопленную сумму на зарплатный счет клиента). Все операции должны сопровождаться необходимыми проверками на указанные выше ограничения.
Класс **ProcCenter** должен использоваться для поддержки работы класса **Depozit** и может быть разработан в минимально-необходимом варианте.

5.4. Кредит и Процессинговый центр

Разработать классы **Credit** и **ProcCenter**.
Класс **Credit** должен имитировать работу интернет-банка в части управления кредитом для зарплатных клиентов банка (тех, чья зарплата перечисляется работодателем на счета работников, открытые в данном банке). Кредит может выдаваться клиенту на выбранный им срок. Возможные варианты: 1 год, 2 года, 3 года, 5 лет, 15 лет. Считать, что процентные ставки по кредитам зависят только от срока и суммы кредита. Диапазоны сумм: до 100 тыс. рублей, от 100 до 500 тыс. рублей, от 500 тыс. до 1 млн рублей, от 1 до 3 млн рублей. Процентные ставки указываются за 1 год (например, 12% годовых по кредиту на 3 года на сумму от 500 тыс. до 1 млн рублей или 15% годовых по кредиту на 1 год на сумму от 100 до 500 тыс. рублей). Выплаты по кредиту начисляются клиенту раз в месяц. Клиент может выплатить сумму больше или равную начисленной. Клиент может досрочно погасить кредит. Считать, что кредит одобряется банком клиенту, если текущая сумма на его зарплатном счету достаточна для шести ежемесячных выплат по кредиту. После одобрения банком сумма кредита перечисляется на зарплатный счет клиента.
База клиентов хранится в классе **ProcCenter**. Считать, что информация о клиенте состоит из номера зарплатного счета (для упрощения – номер счета от «0001» до

«9999»), ФИО владельца, суммы на зарплатном счету, (для упрощения – без копеек), информации о кредите, пароля (произвольная строка, выбранная пользователем, с длиной больше 3 символов). Считать, что авторизация клиента в интернет-банке происходит по номеру зарплатного счета и паролю.
Класс **Credit** должен предоставлять следующие операции: 1) авторизовать клиента, 2) показать информацию о доступных клиенту кредитах 3) проверить наличие взятого в банке кредита, 4) проверить возможность получения выбранного кредита, 5) получить выбранный кредит, 6) показать текущее состояние кредита, 7) погасить начисления по кредиту, 8) досрочно погасить кредит.
Все операции должны сопровождаться необходимыми проверками на указанные выше ограничения.
Класс **ProcCenter** должен использоваться для поддержки работы класса **Credit** и может быть разработан в минимально-необходимом варианте.

5.5. Билетная касса Кинотеатр

Разработать классы **TicketOffice** и **Cinema**
Класс **Билетная касса** должен имитировать работу кассы по продаже билетов на киносеансы в многозальном кинотеатре. Считать, что продажа билетов проводится на сеансы в пределах трех дней от текущей даты. Каждый сеанс описывается датой, временем начала сеанса, названием фильма, номером зала, стоимостью билета в зависимости от зоны (VIP и обычная). Для упрощения считать, что покупатель указывает тип зоны и требуемое число билетов, а места выделяются кассой автоматически (при наличии свободных). Зрительные места в каждом зале описываются номером ряда и номером в ряду. Для упрощения считать, что число мест во всех рядах в одном зале одинаково. Продажа билетов на сеанс прекращается через 10 минут после начала сеанса.
Информация о всех сеансах на ближайшие 30 дней проката хранится в классе **Кинотеатр**. Для каждого зала установлена базовая стоимость билетов (на дневные сеансы – от 12.00 до 18.00). Стоимость билетов на утренние сеансы (до 12.00) составляет 75% от базовой, стоимость билетов на вечерние сеансы (после 18.00) – 150% от базовой. Информация о зрительных местах (свободно/занято) в каждом зале на каждом сеансе также хранится в классе **Кинотеатр**.
Класс **Билетная касса** должен предоставлять следующие операции: 1) принять заказ билетов покупателя: дату, время сеанса, название фильма, номер зала, тип зоны, число мест, 2) проверить наличие требуемого количества свободных мест в требуемой зоне, 3) зарезервировать требуемое количество мест, 4) рассчитать общую стоимость билетов, 5) отменить заказ билетов, 6) сформировать билеты (каждый билет включает: дату, время сеанса, название фильма, номер зала, номер ряда, номер места в ряду).
Класс **Кинотеатр** должен использоваться для поддержки работы класса **Билетная касса** и может быть разработан в минимально-необходимом варианте.

5.6. Железнодорожная касса и Горьковская железная дорога

Разработать классы **RailTicketOffice** и **GorkyRailway**.
Класс **Железнодорожная касса** должен имитировать работу кассы по продаже билетов на поезда Нижний Новгород – Москва.

- Считать, что продажа билетов проводится на поезда в пределах 30 дней от текущей даты.

- Считать, что по маршруту Нижний Новгород – Москва курсирует три скоростных поезда «Ласточка», один фирменный и один скорый поезд в сутки в каждом направлении.
- Все вагоны в поездах «Ласточка» однотипны и содержат по 100 сидячих мест.
- В фирменном и скором поездах вагоны трех типов: плацкартные (27 верхних, 27 нижних мест), купейные (18 верхних, 18 нижних мест), СВ (18 нижних мест).
- Число вагонов в поездах «Ласточка» – 8. В фирменном поезде – 2 вагона СВ, 6 купейных вагонов, 4 плацкартных вагона. В скором поезде – 4 купейных вагона, 8 плацкартных вагонов.
- Поезда идентифицируются номерами (десять номеров из диапазона от 1 до 100), вагоны – номерами (целые числа от 1 до 12), места – номерами (целые числа от 1 до максимума для данного типа вагона).
- Информация о всех поездах и всех проданных билетах хранится в классе **Горьковская железная дорога**. Для каждого поезда, каждого типа вагона и каждого типа места установлена стоимость билета. Считать, что все поезда не делают промежуточных остановок по маршруту.

Класс **Железнодорожная касса** должен предоставлять следующие операции: 1) принять заказ билетов покупателя: дату, поезд, тип вагона (если есть выбор), количество билетов каждого возможного вида (если есть выбор), ФИО пассажиров 2) проверить наличие свободных мест по запросу покупателя (при невозможности выдать все билеты в одном вагоне, считать заказ невыполнимым), 3) зарезервировать места, 4) рассчитать общую стоимость билетов, 5) отменить заказ билетов, 6) сформировать билеты (каждый билет включает: дату, номер поезда, номер вагона, номер места, ФИО пассажира, станция отправления, станция прибытия).

Класс **Горьковская железная дорога** должен использоваться для поддержки работы класса **Железнодорожная касса** и может быть разработан в минимально-необходимом варианте.

Задача (Task) 5_1. Варианты задачи

5.1. Касса и Склад

Разработать классы **Kassa** и **Sklad**.
Класс **Касса** должен имитировать работу кассового аппарата по сканированию товаров и формированию чека за покупку. Каждый товар идентифицируется штрих-кодом (для упрощения – строка из четырех цифр от «0001» до «9999»). Один и тот же товар может сканироваться несколько раз, но в чек информация о каждом товаре входит в виде «наименование – стоимость за единицу (для упрощения в рублях без копеек) – количество – общая стоимость за товар». Чек состоит не менее чем из одной записи указанного вида. Чек дополнительно включает общую стоимость товаров в покупке, суммарную скидку и итоговую сумму к оплате (все в рублях).
База товаров хранится в классе **Склад**. Товар описывается штрих-кодом, наименованием, стоимостью за единицу товара, скидкой в процентах от стоимости. Скидки устанавливаются на каждый товар независимо (в диапазоне от 1 до 50%).
Класс **Касса** должен предоставлять следующие операции: 1) «сканировать» очередной товар, 2) получить описание товара со склада, 3) добавить данные о товаре в чек, 4) сформировать чек за покупку, 5) рассчитать итоговую сумму к оплате, 6) удалить выбранный товар из покупки.
Класс **Склад** должен использоваться для поддержки работы класса **Касса** и может быть разработан в минимально-необходимом варианте.

5.2. Банкомат и Процессинговый центр

Разработать классы **Bankomat** и **ProcCenter**.
Класс **Bankomat** должен имитировать работу банкомата по приему и выдаче наличных денежных средств (в рублях).
Класс должен поддерживать работу с купюрами в 100, 200, 500, 1000, 2000, 5000 рублей. Купюры каждого достоинства хранятся в отдельной кассете. Емкость каждой кассеты – 2000 купюр. Считать, что начальная загрузка банкомата составляет 80% для каждой кассеты.
Выдаваемая или принимаемая за одну операцию сумма ограничена 40 купюрами (независимо от их достоинства). Считать, что клиент банка идентифицируется по номеру пластиковой карты (для упрощения – номер карты от «0001» до «9999»).
База клиентов хранится в классе **ProcCenter**. Номера выданных клиентам карт могут идти не подряд. Считать, что информация о клиенте состоит из номера карты, ФИО владельца, суммы на счету, (для упрощения – без копеек), PIN-кода (последовательность из 4-х цифр, каждая от 0 до 9).
Класс **Bankomat** должен предоставлять следующие операции: 1) принять карту клиента, 2) найти клиента по номеру карты, 3) проверить PIN-код, 4) распечатать состояние счета клиента, 5) выдать клиенту наличные (списав выданную сумму со счета), 6) принять от клиента наличные (зачислив полученную сумму на счет), 7) заблокировать карту клиента, если до ее возврата клиенту три раза подряд набран неверный PIN-код, 8) вернуть карту клиенту.
Все операции должны сопровождаться необходимыми проверками на указанные выше ограничения.
Класс **ProcCenter** должен использоваться для поддержки работы класса **Bankomat** и может быть разработан в минимально-необходимом варианте.

5.3. Депозит и Процессинговый центр

Разработать классы **Depozit** и **ProcCenter**.
Класс **Depozit** должен имитировать работу интернет-банка в части управления депозитом для зарплатных клиентов банка (тех, чья зарплата перечисляется работодателем на счета работников, открытые в данном банке). Депозит может открываться клиентом на выбранный срок. Возможные варианты: 3 месяца, 6 месяцев, 1 год, 2 года, 3 года. Считать, что процентные ставки по депозитам зависят только от срока депозита и начальной суммы. Диапазоны начальных сумм: до 100 тыс. рублей, от 100 до 500 тыс. рублей, от 500 тыс. до 1 млн рублей, свыше 1 млн рублей. Процентные ставки указываются для срока 1 год (например, 6% годовых по вкладу на 6 месяцев от 100 до 500 тыс. рублей или 6.6% годовых по вкладу на 1 год от 500 тыс. до 1 млн рублей). Считать, что проценты начисляются раз в месяц на начальную сумму депозита и могут быть сняты клиентом в любой момент. Депозит может быть закрыт только по истечении срока, на который он был открыт.
База клиентов хранится в классе **ProcCenter**. Считать, что информация о клиенте состоит из номера зарплатного счета (для упрощения – номер счета от «0001» до «9999»), ФИО владельца, суммы на зарплатном счету, (для упрощения – без копеек), информации о депозите, пароля (произвольная строка, выбранная пользователем, с длиной больше 3 символов). Считать, что авторизация клиента в интернет-банке происходит по номеру зарплатного счета и паролю.
Класс **Depozit** должен предоставлять следующие операции: 1) авторизовать клиента, 2) показать информацию о доступных клиенту депозитах, исходя из суммы на его зарплатном счету, 3) проверить наличие открытого депозита, 4) открыть депозит (переведя указанную сумму с зарплатного счета клиента на депозит), 5) показать состояние депозита, 6) снять проценты (переведя их на зарплатный счет клиента), 7) закрыть депозит (переведя всю накопленную сумму на зарплатный счет клиента). Все операции должны сопровождаться необходимыми проверками на указанные выше ограничения.
Класс **ProcCenter** должен использоваться для поддержки работы класса **Depozit** и может быть разработан в минимально-необходимом варианте.

5.4. Кредит и Процессинговый центр

Разработать классы **Credit** и **ProcCenter**.
Класс **Credit** должен имитировать работу интернет-банка в части управления кредитом для зарплатных клиентов банка (тех, чья зарплата перечисляется работодателем на счета работников, открытые в данном банке).
Кредит может выдаваться клиенту на выбранный им срок. Возможные варианты: 1 год, 2 года, 3 года, 5 лет, 15 лет.
Считать, что процентные ставки по кредитам зависят только от срока и суммы кредита. Диапазоны сумм: до 100 тыс. рублей, от 100 до 500 тыс. рублей, от 500 тыс. до 1 млн рублей, от 1 до 3 млн рублей. Процентные ставки указываются за 1 год (например, 12% годовых по кредиту на 3 года на сумму от 500 тыс. до 1 млн рублей или 15% годовых по кредиту на 1 год на сумму от 100 до 500 тыс. рублей). Выплаты по кредиту начисляются клиенту раз в месяц. Клиент может выплатить сумму больше или равную начисленной. Клиент может досрочно погасить кредит. Считать, что кредит одобряется банком клиенту, если текущая сумма на его зарплатном счету достаточна для шести ежемесячных выплат по кредиту. После одобрения банком сумма кредита перечисляется на зарплатный счет клиента.

База клиентов хранится в классе **ProcCenter**. Считать, что информация о клиенте состоит из

- номера зарплатного счета (для упрощения – номер счета от «0001» до «9999»),
- ФИО владельца,
- суммы на зарплатном счету, (для упрощения – без копеек),
- информации о кредите,
- пароля (произвольная строка, выбранная пользователем, с длиной больше 3 символов).

Считать, что авторизация клиента в интернет-банке происходит по номеру зарплатного счета и паролю.
Класс **Credit** должен предоставлять следующие операции:
1) авторизовать клиента,
2) показать информацию о доступных клиенту кредитах
3) проверить наличие взятого в банке кредита,
4) проверить возможность получения выбранного кредита,
5) получить выбранный кредит,
6) показать текущее состояние кредита,
7) погасить начисления по кредиту,
8) досрочно погасить кредит.
Все операции должны сопровождаться необходимыми проверками на указанные выше ограничения.
Класс **ProcCenter** должен использоваться для поддержки работы класса **Credit** и может быть разработан в минимально-необходимом варианте.

5.5. Билетная касса Кинотеатр

Разработать классы **TicketOffice** и **Cinema**
Класс **Билетная касса** должен имитировать работу кассы по продаже билетов на киносеансы в многозальном кинотеатре. Считать, что продажа билетов проводится на сеансы в пределах трех дней от текущей даты.
Каждый сеанс описывается

- датой,
- временем начала сеанса,
- названием фильма,
- номером зала,
- стоимостью билета в зависимости от зоны (VIP и обычная).

Для упрощения считать, что покупатель указывает тип зоны и требуемое число билетов, а места выделяются кассой автоматически (при наличии свободных). Зрительные места в каждом зале описываются номером ряда и номером в ряду. Для упрощения считать, что число мест во всех рядах в одном зале одинаково. Продажа билетов на сеанс прекращается через 10 минут после начала сеанса.
Информация о всех сеансах на ближайшие 30 дней проката хранится в классе **Кинотеатр**. Для каждого зала установлена базовая стоимость билетов (на дневные сеансы – от 12.00 до 18.00). Стоимость билетов на утренние сеансы (до 12.00) составляет 75% от базовой, стоимость билетов на вечерние сеансы (после 18.00) – 150% от базовой. Информация о зрительных местах (свободно/занято) в каждом зале на каждом сеансе также хранится в классе **Кинотеатр**.
Класс **Билетная касса** должен предоставлять следующие операции:
1) принять заказ билетов покупателя: дату, время сеанса, название фильма, номер зала, тип зоны, число мест,

2) проверить наличие требуемого количества свободных мест в требуемой зоне,
3) зарезервировать требуемое количество мест,
4) рассчитать общую стоимость билетов,
5) отменить заказ билетов,
6) сформировать билеты (каждый билет включает: дату, время сеанса, название фильма, номер зала, номер ряда, номер места в ряду).
Класс **Кинотеатр** должен использоваться для поддержки работы класса **Билетная касса** и может быть разработан в минимально-необходимом варианте.

5.6. Железнодорожная касса и Горьковская железная дорога

Разработать классы **RailTicketOffice** и **GorkyRailway**.
Класс **Железнодорожная касса** должен имитировать работу кассы по продаже билетов на поезда Нижний Новгород – Москва.

- Считать, что продажа билетов проводится на поезда в пределах 30 дней от текущей даты.
- Считать, что по маршруту Нижний Новгород – Москва курсирует три скоростных поезда «Ласточка», один фирменный и один скорый поезд в сутки в каждом направлении.
- Все вагоны в поездах «Ласточка» однотипны и содержат по 100 сидячих мест.
- В фирменном и скором поездах вагоны трех типов: плацкартные (27 верхних, 27 нижних мест), купейные (18 верхних, 18 нижних мест), СВ (18 нижних мест).
- Число вагонов в поездах «Ласточка» – 8. В фирменном поезде – 2 вагона СВ, 6 купейных вагонов, 4 плацкартных вагона. В скором поезде – 4 купейных вагона, 8 плацкартных вагонов.
- Поезда идентифицируются номерами (десять номеров из диапазона от 1 до 100), вагоны – номерами (целые числа от 1 до 12), места – номерами (целые числа от 1 до максимума для данного типа вагона).
- Информация о всех поездах и всех проданных билетах хранится в классе **Горьковская железная дорога**. Для каждого поезда, каждого типа вагона и каждого типа места установлена стоимость билета. Считать, что все поезда не делают промежуточных остановок по маршруту.

Класс **Железнодорожная касса** должен предоставлять следующие операции: 1) принять заказ билетов покупателя: дату, поезд, тип вагона (если есть выбор), количество билетов каждого возможного вида (если есть выбор), ФИО пассажиров 2) проверить наличие свободных мест по запросу покупателя (при невозможности выдать все билеты в одном вагоне, считать заказ невыполнимым), 3) зарезервировать места, 4) рассчитать общую стоимость билетов, 5) отменить заказ билетов, 6) сформировать билеты (каждый билет включает: дату, номер поезда, номер вагона, номер места, ФИО пассажира, станция отправления, станция прибытия).
Класс **Горьковская железная дорога** должен использоваться для поддержки работы класса **Железнодорожная касса** и может быть разработан в минимально-необходимом варианте.

Задача (Task) 6. Варианты задачи

6.1. Быки и коровы

Разработать систему классов и реализовать с ее помощью игру «Быки и коровы».
Требования (правила).

11) Играют два игрока (человек и компьютер).
12) Игрок выбирает длину загадываемого числа – n.
13) Компьютер «задумывает» n-значное число с неповторяющимися цифрами.
14) Игрок делает попытку отгадать число – вводит n-значное число с неповторяющимися цифрами.
15) Компьютер сообщает, сколько цифр угадано без совпадения с их позициями в загаданном числе (то есть количество коров) и сколько угадано вплоть до позиции в загаданном числе (то есть количество быков).
16) Игрок делает попытки, пока не отгадает всю последовательность.

Пример.

17) Пусть n = 4.
18) Пусть задумано тайное число «3219».
19) Игрок ввел число «2310».
20) Результат: две «коровы» (две цифры: «2» и «3» — угаданы на неверных позициях) и один «бык» (одна цифра «1» угадана вплоть до позиции).

6.2. Змейка

Разработать систему классов и реализовать с ее помощью игру «Змейка».
Требования (правила).

21) Играет один игрок (человек), управляющий «змейкой».
22) Игра идет на прямоугольном поле N x M клеток. Поле ограничено «стенами» так что вместе со стенами размер поля – (N + 2) x (M + 2) клеток.
23) При старте игры змейка имеет длину 5 клеток, форму в виде горизонтального отрезка и располагается в произвольном месте поля, не пересекая и не касаясь стен.
24) При старте игры «голова» змейки располагается слева, «хвост» справа. Голова змейки окрашена в цвет, отличный от цвета остальных клеток ее тела.
25) При старте игры в произвольной клетке поля (не совпадающей с клетками, занятыми змейкой) возникает «пища».
26) При запуске игрового процесса (по специальной команде или автоматически при старте игры) змейка начинает автоматическое движение влево с некоторой заданной скоростью.
27) Движение заключается в том, что за каждый такт голова змейки перемещается на одну клетку в текущем направлении движения, а клетка, в которой располагался хвост, становится пустой.
28) Игрок может сменить направление движения змейки с помощью клавиш-«стрелок» (вверх, вниз, влево, вправо).
29) Если на текущем такте движения голова змейки должна будет занять клетку стены или клетку, которая уже занята любой из клеток ее тела, игра прекращается и считается проигранной.
30) Задача игрока вырастить змейку до заданного при старте игрового процесса размера. Змейка вырастает в длину на одну клетку (с хвоста) при каждом поглощении пищи, т.е. в тот момент, когда ее голова на очередном такте движения занимает клетку, в которой расположена пища. На этом же такте в произвольном месте игрового поля (не совпадающей с клетками, занятыми змейкой) снова появляется пища.

31) Если змейка выросла до заданной при старте игрового процесса длины, игра считается выигранной.

6.3. Морской бой

Разработать систему классов и реализовать с ее помощью игру «Морской бой».
Требования (правила).

32) Играют два игрока (человек и компьютер).
33) У каждого игрока два поля 10x10 клеток. В левом поле игрок расставляет свои корабли. В правом игрок пытается потопить чужие корабли.
34) У каждого игрока имеются 4 «однопалубных» (из одной клетки) корабля, 3 «двухпалубных», 2 «трехпалубных» и 1 «четырехпалубный» корабль.
35) Многопалубные корабли могут располагаться только по горизонтали или вертикали.
36) Корабли не могут располагаться в соседних клетках. Соседними для каждой клетки считаются 8 окружающих ее клеток.
37) Игра состоит из поочередных ходов игроков.
38) Первый ход выполняет человек.
39) Каждый ход состоит из следующих действий:
    a. Игрок, выполняющий ход, «называет» выбранную клетку (координаты).
    b. Соперник проверяет «попадание» на своей доске. Если в названной клетке расположен корабль, соперник оглашает попадание, иначе промах.
    c. Игрок, выполняющий ход, ставит на своей правой доске по названным координатам отметку о результатах хода.
    d. Если игрок, выполняющий ход, попал в корабль, ход остается у него, иначе переходит к сопернику.
40) Выигрывает тот игрок, кто первым потопит все корабли противника.