Продвинутые темы и техники C++

Урок 1. Чтение из файлов

**Цели практической работы**

Приобрести практическое понимание работы с файлами.  
Научиться:  
• открывать файлы по указанному пути,  
• считывать разные типы данных из текстовых файлов,  
• считывать данные из бинарных файлов,  
• составить простейшие текстовые данные в редакторе,  
• считывать массивы данных.  
  
**Что входит в практическую работу**  
1. Разработать программу поиска слов в файле.  
2. Разработать простейший просмотрщик текстовых файлов.  
3. Реализовать программу чтения и анализа ведомости с выплатами.  
4. Разработать детектор PNG-файла.  
5. Реализовать версию игры «Что? Где? Когда?».

**Общие условия**

Разработка выполняется в среде [VS Code](https://code.visualstudio.com/download). Текстовые документы можно редактировать в стандартном блокноте.

Задание 1. Разработка программы поиска слов в файле

**Что нужно сделать**  
Составьте небольшой текстовый файл «words.txt», в котором последовательно, через пробелы и переносы строк расположены различные слова. Можете использовать любые англоязычные слова. Разработайте программу, которая искала бы среди этих слов нужное пользователю и выводила бы общее количество повторений этого слова. 0 — если слово вообще не присутствует в файле.

Задание 2. Разработка просмотрщика текстовых файлов

**Что нужно сделать**  
Разработайте простейший просмотрщик текстовых файлов. В начале программы пользователь вводит путь к текстовому файлу, который требуется открыть и просмотреть. Внутренности файла необходимо вывести в стандартный вывод. Программа должна работать для любых текстовых файлов в формате TXT, но вы можете создать и свой отдельный файл для теста. Главное, чтобы файл был англоязычным, дабы избежать проблем с кодировками.  
  
**Советы и рекомендации**  
• Для проверки того, был ли на самом деле открыт файл, используйте метод is\_open. Он вернёт true, если файл был благополучно открыт, и false, если что-то пошло не так, к примеру, если путь к файлу невалидный.  
• Файл надо считывать постепенно, отдельными фрагментами, ведь он может быть весьма большим. Для этого лучше всего пользоваться методом gcount, так как он ещё вернёт количество байт, которые были считаны по факту методом read. Сами эти фрагменты могут храниться в специальном буфере, как и в примере из лекций, но его размер и второй аргумент к функции read надо рассчитать так, чтобы выводились все символы, без обрезки последней буквы.  
• Файл можно открыть в бинарном формате.

Задание 3. Реализация программы чтения ведомости

**Что нужно сделать**  
Реализуйте программу чтения информации из ведомости по учёту выплат. Сама ведомость представляет собой следующую таблицу:  
• имя и фамилия человека;  
• количество денег, которые были выплачены лицу;  
• дата выплаты в формате ДД.ММ.ГГГГ.  
  
На основе данных из ведомости следует подсчитать общее количество выплаченных средств и определить человека с максимальной суммой выплат.  
Пример данных ведомости  
  
Tom Hanks 35500 10.11.2020  
Rebecca Williams 85000 1.1.2021  
Sally Field 15600 15.8.2021  
Michael Humphreys 29400 23.5.2020  
Harold Herthum 74300 9.6.2019  
George Kelly 45000 12.3.2018  
Bob Penny 12500 13.5.2020  
John Randall 23400 2.10.2020  
Sam Anderson 6500 15.7.2020  
Margo Moorer 12350 24.2.2019ш

Задание 4. Разработка детектора PNG-файла

**Что нужно сделать**  
Разработайте программу, которая получает на вход, в стандартный ввод, путь к файлу. На выходе программа должна определить, является ли данный файл PNG-изображением. Данная процедура не просто должна смотреть на расширение файла, но и произвести минимальный анализ его внутренностей, в бинарном режиме.  
Внутренности валидного бинарного файла начинаются со специального 8-байтового заголовка. Нас будут интересовать первые четыре байта. Первый байт всегда имеет значение −119 (число со знаком минус), а следующие — это просто символы ‘P’, ‘N’, ‘G’, расположенные друг за другом. Обратите внимание, что все они в верхнем регистре.  
По результатам проверки пути и внутренностей требуется сообщить пользователю о результате проверки в стандартный вывод.  
  
**Советы и рекомендации**  
Чтобы определить расширение файла, используйте функцию substr, которая получает офсет и длину подстроки и возвращает её.

Задание 5. Реализация игры «Что? Где? Когда?»

**Что нужно сделать**  
Реализуйте простую версию интеллектуальной игры «Что? Где? Когда?».  
Как происходит игра  
В начале каждого хода игроки вращают волчок, то есть выбирают сектор на столе, который сейчас играет. Всего таких секторов 13. Сектор выбирается так: с клавиатуры вводится офсет (смещение) относительно текущего сектора на барабане. Исходя из этого офсета вычисляется новый активный сектор, который и будет играть в этом ходе. Если выпавший сектор уже играл, выбирает следующий не игравший за ним. В начале всей игры стрелка установлена на первом секторе.  
  
Как только играет какой-то из секторов, с него берётся письмо с вопросом — то есть считывается из файла данного сектора. Вопрос показывается на экране.  
После того как вопрос показан, от игрока ожидается ответ на него. Игрок вводит этот ответ с помощью стандартного ввода. То, что он ввёл, сравнивается с ответом, который хранится во втором файле, ассоциированном с активным сектором. Данный файл должен содержать лишь одно слово-ответ.  
  
Если ответ знатока-игрока был правильным, ему начисляется один балл. Если неверен, то балл уходит телезрителям.  
Игра продолжается до тех пор, пока или игрок, или зрители не наберут шесть баллов. После этого называется победитель и программа заканчивает работу.

### Урок 2. Запись в файлы

#### ****Цели домашнего задания****

Приобрести практическое понимание работы с файлами.  
Научиться:  
• открывать файлы на запись по указанному пути и закрывать их;  
• записывать разные типы данных в файлы;  
• записывать массивы данных в файлы;  
• пользоваться генератором случайных чисел.

#### Задание 1. Реализация записи в ведомость учёта

**Что нужно сделать**  
В одном из прошлых заданий мы уже создавали программу чтения и анализа ведомости выплат. Теперь требуется создать простую программу записи в эту ведомость.  
Формат ведомости прежний. Сначала идёт имя и фамилия получателя денежных средств, далее располагается дата выдачи в формате ДД.ММ.ГГГГ. Завершает запись сумма выплаты в рублях. Данные разделяются между собой пробелами. В конце каждой записи должен быть расположен перевод строки.  
При старте программы пользователь последовательно вводит данные для новой записи, которые записываются затем в файл в текстовом режиме. Программа должна добавлять новые записи в конец файла, не удаляя его текущее содержимое.  
Советы и рекомендации  
Введённые данные рекомендуется хотя бы минимально валидировать перед их записью.  
Для валидации даты в указанном формате можно использовать функции std::stoi и std::subtr с корректно указанным диапазоном символов в дате. К примеру, вызов std::stoi(date.substr(3, 2)) вычленит из строковой даты целочисленный месяц для последующего анализа.  
Что оценивается  
Корректность работы программы. Программа должна дозаписывать данные в конец существующего списка.  
Как отправить задание на проверку  
Прислать ссылку на repl.it или файл .срр с решением.

#### Задание 2. Реализация рисования случайных картин

**Что нужно сделать**  
С помощью генератора случайных чисел рисуется картинка из нулей и единиц. На вход от пользователя принимается размер картины: высота и ширина в пикселях. На выходе нужно создать файл pic.txt, содержимое которого будет наполнено случайными нулями и единицами в рамках размера картины.  
Это ещё не всё. Если вы запустите программу несколько раз на одних и тех же исходных данных (ширина и высота картины), вы очень скоро обнаружите, что всё время выводится одна и та же картинка. Дело в том, что у генератора случайных чисел есть ещё и «зерно», от которого он отталкивается. Если это зерно не меняется, то и серия из rand() всегда будет возвращать одни и те же случайные числа.  
Чтобы разнообразить вывод, вы можете использовать вызов std::srand(std::time(nullptr)); в самом начале программы. Эта операция задаёт начальное зерно случайных чисел в зависимости от текущего времени. Для её корректной работы не забудьте также включить заголовочный файл <ctime>.  
Что оценивается  
Корректность работы программы по созданию случайных изображений заданного размера.  
Как отправить задание на проверку  
Прислать ссылку на repl.it или файл .срр с решением.

#### Задание 3. Реализация симуляции игры «Рыбалка»

**Что нужно сделать**  
Реализуйте простую симуляцию игры в рыбалку. В исходном текстовом файле реки (river.txt) задаётся список из видов рыб, которые можно в ней поймать. Рыбок может быть сколько угодно, разных видов. Виды при этом могут повторяться.  
В начале программы пользователь указывает, какую именно рыбу он сейчас будет ловить — вид этой рыбы. После этого из первого файла друг за другом осуществляется чтение его содержимого, вид за видом. Если на отдельных шагах вид совпал с указанным пользователем, в выходной файл basket.txt (корзинка) записывается этот вид.  
В конце программы показывается, сколько было поймано рыб за текущую ловлю. Программу можно запускать несколько раз, при этом уже пойманные рыбы должны сохраняться в файле-корзинке.  
Пример содержания исходного файла  
  
sunfish  
shad  
carp  
bass  
bullhead  
carp  
walleye  
catfish  
carp

#### Задание 4. Реализация симуляции банкомата

**Что нужно сделать**  
Требуется реализовать упрощённую симуляцию работы банкомата. В банкомате могут храниться только бумажные купюры номиналом от 100 до 5000 рублей. Максимально в банкомате может храниться только 1000 купюр. Все они записываются в отдельный файл. У банкомата, как у устройства, две основных операции — снятие денег пользователем и наполнение деньгами инкассаторами банка.  
  
Наполнение банкомата происходит, если при старте программы в консоль вводится символ плюс “+”. Количество купюр рассчитывается так, чтобы банкомат был заполнен полностью. Все купюры при этом выбираются случайным образом.  
Если на старте программы в консоль вводится минус (“-”), то симулируется снятие пользователем денег. Пользователь указывает сумму с точностью до 100 рублей. Мы будем считать, что каждый клиент обладает неограниченным балансом в системе и теоретически может снять любую сумму. На практике, если данная сумма не может быть снята из-за отсутствия подходящих денег в машине, показывается сообщение, что эта операция невозможна.  
После выполнения любой из операций программа завершает выполнение. Состояние банкомата должно храниться в отдельном бинарном файле, автономно.  
  
**Советы и рекомендации**  
Вы можете хранить массив купюр целиком, помечая отсутствующие позиции нулями.

### Урок 8. Введение в ООП. Часть 1

**Цели практической работы**  
1. Получить первичный практический навык работы с классами в C++.  
2. Научиться:  
— объявлять классы,  
— пользоваться типами доступа,  
— определять методы класса.

#### Задание 1. Реализация имитации аудиоплеера

**Что нужно сделать**  
С помощью классов реализуйте программу имитации работы аудиоплеера. В плеере находится несколько аудиозаписей, доступных для воспроизведения. Для них вы должны создать отдельный класс Track. Полями этого класса должны быть: название, дата создания (здесь вы можете использовать тип std::tm) и продолжительность записи в секундах.  
Сам плеер тоже реализуйте с помощью класса. Внутри он должен содержать список доступных записей и нужные для логики работы плеера поля.  
  
Пользователь взаимодействует с программой с помощью команд:  
  
• Команда **play** запрашивает у пользователя название записи для воспроизведения и начинает её воспроизведение. Воспроизведение симулируется с помощью вывода в консоль всей информации о записи. Если произведение какой-то записи уже начато, ничего не происходит.  
• Команда **pause**. Ставит запись на паузу, о чём выводит соответствующее сообщение в консоль. Если запись уже на паузе или не воспроизводится вовсе, ничего не происходит.  
• Команда **next** переходит к следующей записи для воспроизведения. Запись при этом выбирается случайно, как будто плеер работает в режиме shuffle.  
• Команда **stop** прекращает воспроизведение текущей записи, если таковая имеется. Об этом выводится сообщение в консоль. Два раза остановить воспроизведение невозможно, так что команда должна срабатывать только в первом случае.  
• Команда **exit** — выход из программы.  
  
**Советы и рекомендации**  
Забудьте про отдельные глобальные функции и используйте возможности классов и методов по максимуму.

#### Задание 2. Реализация программы симуляции мобильного телефона

**Что нужно сделать**  
Реализуйте программу простой симуляции работы мобильного телефона.  
По мобильному телефону можно звонить и отправлять СМС. Также мобильный телефон содержит адресную книгу. Телефон и отдельные записи адресной книги должны быть реализованы с помощью классов. Все номера телефонов задаются в формате +7 <10 цифр>.  
  
Пользователь взаимодействует с программой с помощью команд:  
  
• Команда **add** добавляет в адресную книгу новый номер телефона и ассоциирует его с именем контакта. Оба эти параметра команда получает от пользователя через стандартный ввод.  
• Команда **call** запрашивает у пользователя имя контакта или номер телефона, на который требуется позвонить. Сам по себе звонок симулируется с помощью вывода в консоль сообщения CALL с номером телефона, на который осуществляется вызов.  
• Команда **sms**. Запрашивает у пользователя номер телефона (или имя контакта), на который требуется послать сообщение. Само сообщение также вводится через стандартный ввод.  
• Команда **exit** — выход из программы.  
  
**Советы и рекомендации**  
Сам по себе тип номера телефона вы тоже можете реализовать с помощью отдельного класса.

#### Задание 3. Реализация программы управления окном рабочего стола

**Что нужно сделать**  
Реализуйте симуляцию работы окна в визуальном интерфейсе пользователя.  
Окно располагается на виртуальном экране монитора, размером 80 × 50 точек. У окна есть несколько параметров, которые также задаются в пикселях: двумерные координаты левого верхнего угла и размеры, ширина и высота. Ширина и высота при этом не могут быть отрицательными, а координаты не могут выходить за область экрана.  
  
Пользователь взаимодействует с программой с помощью команд:  
  
• Команда **move** принимает вектор, на который надо сместить окно на экране. Сообщение с новыми координатами окна появляется на экране.  
• Команда **resize** запрашивает из стандартного ввода новый размер окна. После изменения размера сообщение с новым размером выводится на экране.  
• Команда **display** выводит текущее изображение монитора в консоль. 0 — это пиксель вне области окна, 1 — с окном.  
• Команда **close** закрывает окно, выход из программы.  
  
**Советы и рекомендации**  
Сам экран монитора вы тоже можете реализовать с помощью класса.

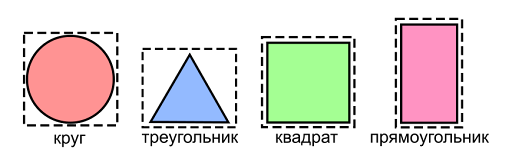
### Урок 9. Введение в ООП. Часть 2

**Цели практической работы**  
1. Получить первичный практический навык работы с классами в C++.  
2. Научиться:  
— создавать и вызывать конструкторы,  
— пользоваться указателем this,  
— выстраивать иерархии классов.

#### Задание 1. Реализация деревни эльфов

**Что нужно сделать**  
Лесные эльфы расположили свою деревню в лесу, прямо на деревьях. Нужно расселить эльфов по веткам деревьев, а затем подсчитать общее количество соседей определённого эльфа.  
Всего в лесу пять деревьев, каждое из которых генерируется с помощью случайных чисел, заданных в указанных далее диапазонах. На каждом дереве есть 3–5 основных больших ветвей. На каждой большой ветке расположены ещё 2-3 средние ветки. Дома эльфов построены на больших и на средних ветках. Более мелкие ветви не рассматриваются в этой задаче.  
В начале программы пользователь размещает в каждом доме по одному эльфу. Делается это с помощью последовательного перебора всех имеющихся домов и запроса имени заселяемого эльфа через стандартный ввод. Если было введено None в качестве имени, то дом пропускается и не заселяется никем.  
После этого требуется найти определённого эльфа по имени. Имя искомого эльфа вводится через стандартный ввод. Для найденного эльфа нужно вывести общее количество эльфов, живущих вместе с ним на одной большой ветви. Это и будет искомое количество соседей.  
  
**Советы и рекомендации**  
И для представления деревьев, и для представления больших и средних ветвей следует использовать один и тот же класс (например Branch) с полем, которое будет обозначать родительскую ветвь, и полем, обозначающим дочерние ветви. У самого дерева, в свою очередь, родительская ветвь будет всегда нулевой, то есть nullptr.  
Функцию поиска эльфа в дереве можно реализовать с помощью рекурсивного метода в классе Branch, который сначала поищет эльфа в себе, а потом уже на дочерних ветвях.  
Чтобы лучше представлять структуру дерева и понимать, как получать его вершину, взгляните на данный метод:  
  
Branch\\* getTopBranch()  
{  
//Если это дерево  
if (parent == nullptr) return nullptr;  
//Если это средняя ветка  
if (parent->parent == nullptr) return parent;  
//Если это маленькая ветка  
return parent->getTopBranch();  
}  
[]()

#### Задание 2. Иерархия геометрических фигур

**Что нужно сделать**  
Спроектируйте иерархию классов различных фигур: круг, квадрат, равносторонний треугольник, прямоугольник.  
Для всех этих фигур есть общие поля-данные — это координаты их центра и условный цвет фигуры: красный, синий или зелёный.  
Для отдельных фигур есть и уникальные параметры: радиус для круга, длина ребра для квадрата и равностороннего треугольника, ширина и высота для прямоугольника. Все данные — это вещественные числа с удвоенной точностью.  
Для каждой из фигур требуется определить метод нахождения площади фигуры, а также метод нахождения прямоугольника, описывающего фигуру полностью (он может быть больше зоны фигуры, но не меньше).  
Для лучшего понимания задачи приведена иллюстрация. На ней разными цветами помечены площади фигур, а пунктиром изображены прямоугольники, описывающие фигуры.  
  
Для теста предусмотрите ввод пользователем команд: circle, square, triangle, rectangle, соответствующих фигурам. Команды должны получать параметры фигур и выводить их цвет, площадь и описывающий прямоугольник с координатами.  
  
**Советы и рекомендации**  
Для цвета фигуры вы можете использовать собственный тип enum, в котором может содержаться и константа отсутствия цвета — None.  
Площадь треугольника считается как length \* length \* std::sqrt(3) / 4; где length — сторона. Площадь круга — как atan(1) \* 4 \* radius;  
Геометрические формулы вы можете найти в интернете.

#### Задание 3. Симуляция работы компании

**Что нужно сделать**  
Реализуйте симуляцию работы компании. В компании существует вертикальное управление с иерархией. Есть основные рабочие, подчинённые, разбитые на команды. У каждой команды есть менеджер среднего звена. Всей компанией руководит глава компании. У всех сотрудников, включая руководителя, есть имена.  
  
Глава компании выполняет стратегическое управление. Он определяет основной вектор движения — задачи верхнего уровня. Менеджеры среднего звена разбивают эти задачи на подзадачи и раздают их своим подчинённым. Подчинённые выполняют работу.  
• Реализация данной логики должна проходить упрощённым образом. Указания главы компании поступают из стандартного ввода и являются не более чем целочисленными идентификаторами.  
• Данные идентификаторы служат зерном для генератора случайных чисел. Каждый из менеджеров, принимая себе эту целочисленную команду, добавляет к ней свой порядковый номер и вызывает с результирующей суммой функцию std::srand().  
• На основе этой суммы вычисляется общее количество задач, которое требуется выполнить данной командой, — от 1 до количества работников в группе. Всего есть три типа задач для работников: A, B и C. Они тоже выбираются случайно и распределяются между незанятыми рабочими.  
• При старте программы пользователь указывает в стандартном вводе количество команд и количество работников в каждой из команд.  
• Далее пользователь вводит целые числа, которые соответствуют указаниям руководителя компании. Когда какой-то работник или менеджер получил указания или задачу, об этом сообщается в стандартный вывод, включая имя работающего сотрудника. Программа завершается, когда все работники заняты какими-то задачами.  
  
**Советы и рекомендации**  
Пользуйтесь всем изученным в модуле, включая иерархии классов для представления сотрудников и указатель this, если потребуется.  
Чтобы проинициализировать генератор случайных чисел, используйте std::srand(hash); где hash — некое случайное и уникальное значение, например, ID вашей группы и команды. Для получения количества задач используйте полученный генератор случайных чисел так: int tasksCount = rand() % (team->workers.size() + 1);