Задания к курсовой работе по языкам и методам программирования.

Все задания реализуются на языке программирования C++ (стандарт C++14 и выше). Реализованные в заданиях приложения не должны завершаться аварийно; все возникающие исключительные ситуации должны быть перехвачены и обработаны.

Во всех заданиях запрещено пользоваться функциями, позволяющими завершить выполнение приложения из произвольной точки выполнения.

Во всех заданиях при реализации необходимо разделять контексты работы с данными (поиск, сортировка, добавление/удаление, модификация и т. п.) и отправка данных в поток вывода / выгрузка данных из потока ввода.

Во всех заданиях все вводимые (с консоли, файла, командной строки) пользователем данные должны (если не сказано обратное) быть подвергнуты валидации в соответствии с типом валидируемых данных.

Во всех заданиях необходимо контролировать ситуации с невозможностью [пере]выделения памяти; во всех заданиях необходимо корректно освобождать всю выделенную динамическую память.

Все ошибки, связанные с операциями открытия системных ресурсов уровня ОС (файлы, средства синхронизации, etc.), должны быть обработаны; все открытые системные ресурсы должны быть возвращены ОС.

Реализованные компоненты должны зависеть от абстракций, а не от конкретных реализаций абстракций. Для реализованных компонентов должны быть переопределены (либо перекрыты - при обосновании) следующие механизмы классов C++: конструктор копирования, деструктор, оператор присваивания, конструктор перемещения, присваивание перемещением.

К курсовой работе необходимо подготовить пояснительную записку. Требования к оформлению пояснительной записки указаны в приложении 1.

1. Написать программу для интерпретации инструкций элементарного языка программирования. Рассматриваемый язык программирования позволяет оперировать переменными (целочисленного и вещественного типов), имена которых произвольны, и создавать пользовательские функции. Через аргументы командной строки в программу подается файл с инструкциями вида:

```
foo(x, y): ((x*y+2)*(25-x/y))/(3-(x+2*y));
myfoo2(z): z*z+4;
myvar(i)=15;
bg(f)=25;
ccc=myfoo2(bg+myvar)*15+foo(bg*25+(6*myfoo2(myvar-10)));
print ccc;
bg=ccc*myvar;
print;
```

Файл не содержит ошибок. Реализовать считывание файла и выполнение всех инструкций и операции print (вывод на экран либо текущего значения переменной, либо значений всех переменных с указанием имен). Имя переменной может иметь произвольную длину и содержать символы букв и цифр. Переменная создается в момент инициализации, при этом различают целые переменные (i) и вещественные (f). Заглавные и прописные буквы не отождествляются. В конце программы необходимо очистить всю выделенную память. В случае использования в вычислениях не объявленной переменной или функции вывести сообщение об ошибке.

2. Разработайте программу–интерпретатор операций над целочисленными массивами. Программа оперирует целочисленными массивами произвольной длины с именами A, B, …, Z. Система команд данного интерпретатора (прописные и строчные буквы не различаются):
    1) Load A, in.txt; - загрузить в массив A целые числа из файла in.txt.
    2) Save A, in2.txt; - выгрузить элементы массива A в файл in2.txt.
    3) Rand A, count, lb, rb; - заполнить массив A случайными элементами из отрезка $[lb; lr]$ в количестве count штук.
    4) Concat A, b; - объединить два массива A и B, с результат сохранить в массив A.
    5) Free(a); - очистить массив A.
    6) Remove a, 2, 7; - удалить из массива a 7 элементов, начиная с элемента с индексом 2;
    7) Copy A, 4, 10, b; - скопировать из массива A элементы с 4 по 10 и сохранить их в b;
    8) Sort A+(-); - сортировать элементы массива A по возрастанию (убыванию);
    9) Permute A; - переставить элементы массива в случайном порядке.
    10) Intersect A, B; - найти пересечение массивов A и B, результат сохранить в массив A;
    11) Xor A, B; найти симметрическую разность массивов A и B, результат сохранить в массив A;
    12) Stat a; - вывести статистическую информацию о массиве a: размер массива, максимальный и минимальный элемент (и их индексы), наиболее часто встречающийся элемент, среднее значение элементов, максимальное отклонение элементов от среднего значения.
    13) Print a, 4, 16; - вывести на экран элементы массива начиная с 4 и по 16;
    14) Print a, all; - вывести на экран все элементы массива.

Предоставьте текстовый файл с инструкциями для данного интерпретатора. Считается, что ошибок в командах нет, однако внимательно обработайте все ошибки исполнения инструкций.

3. Вам необходимо разработать для некоторой MMORPG систему усиления игровых персонажей с помощью случайных элементов экипировки.

   Каждый персонаж может иметь некоторое количество ячеек для установки усиления (очень часто под ячейками имеют в виду слоты для элементов одежды, например, шлем, наплечники, перчатки, плащ и др.). Для каждого персонажа определены главные его характеристики: здоровье, броня и, в зависимости от класса персонажа: сила, ум, ловкость, и вторичные характеристики: меткость, везение, мастерство. Персонажи классифицируются следующим образом: защитники, целители, бойцы ближнего боя и бойцы дальнего боя. В зависимости от класса персонажа для него имеет значение та или иная главная характеристика. Для защитников – это здоровье, для целителей – ум, для бойцов ближнего боя – ловкость или сила, для бойцов дальнего боя – ум.

   Вторичные характеристики действуют одинаково для всех классов: меткость определяет возможность промаха при атаке, везение определяет возможность нанесения двойного урона, мастерство увеличивает наносимый урон.

   Продумайте и реализуйте закон для расчета урона (или лечения), наносимого персонажем в зависимости от его главных и вторичных характеристик.

   Продумайте и реализуйте для каждого класса персонажа его способности, а также систему генерации усилений. При создании персонажа никаких усилений (экипировки) у него нет. Продумайте возможность генерации необходимых элементов. Обратите внимание что на каждом предмете экипировки могут быть следующие характеристики: здоровье, броня, ум или сила, или ловкость, и одна или две вторичные характеристики.

   Общая характеристика персонажа определяется как суммарная совокупность значений характеристик на всех его элементах экипировки.

   Разрабатывая данную систему, Вам необходимо использовать стандартный функционал STL. При демонстрации работы приложения Вам необходимо показать генерацию элементов экипировки созданного персонажа, наносимый урон способностями в зависимости от текущих характеристик. Также необходимо иметь возможность создать произвольное количество случайных персонажей (со случайной экипировкой).

4. Реализуйте приложение, осуществляющее подмену слов в тексте. На вход программе через аргументы командной строки подаются пути к текстовым файлам, и на основании словаря синонимов, в каждом текстовом файле происходит подмена синонимичного понятия на эталонное. Словарь синонимов – это набор записей вида <эталонное слово> {список синонимов, разделённых запятой}. При обработке текста выполняется поиск слова по словарю: если текущее слово - эталонное, то замену его производить не требуется; если же оно является синонимом, то необходимо выполнить его замену на соответствующее ему эталонное слово. Обработка файлов происходит в двух режимах: автоматическом и обучающем. В автоматическом режиме отсутствующие в словаре слова игнорируются и, затем, все не заменённые слова сохраняются в файл. В обучающем режиме для каждого слова, которого не оказалось в словаре, выполняется его занесение в словарь: либо добавление к списку синонимов существующего эталонного слова, либо создание нового эталонного слова, либо замена существующего эталонного слова. Предусмотрите возможность операции отмены последних N действий (аналог операции «Отменить»; параметр N является аргументом командной строки). При завершении работы приложения сохраняйте словарь синонимов, при запуске загружайте ранее сохраненный словарь. Ваше приложение должно быть устойчивым: ни при каких обстоятельствах оно не должно аварийно завершаться. Для демонстрации работы используйте содержательный словарь синонимов (> 1000 записей), по вашему выбору, входные файлы могут быть с английским текстом или с русским.

5. В файле находятся служебные записи о деятельности сотрудников некоторой фирмы. Записи имеют следующий формат:

Фамилия Имя Отчество
{
Договор №1 нач. 01.01.2018 кон. 02.03.2018 Работа 123 Стоимость 12500;
Договор №12 нач. 18.04.2019 кон. 20.06.2019 Работа 1234 Стоимость 1550;
…
Договор №14 нач. 19.05.2019 кон. нв Работа 1235 Стоимость 17500;
}

Количество записей не фиксировано, количество информационных полей в каждой записи также не фиксировано. Реализуйте приложение, которое обрабатывает этот текстовый файл, формируя ассоциативный контейнер, который содержит указатели на записи, считанные из файла. Ключевым набором полей каждой записи является ФИО. Ваша программа должна предоставить возможность для заданного сотрудника, определяемого ключевым набором полей:

- подсчёта стоимости всех договоров этого сотрудника;
- выдачи списка договоров, с которыми работал заданный сотрудник;
- найти самый продолжительный договор данного сотрудника;
- найти самый дорогой договор данного сотрудника.

Необходимо также реализовать удаление информации о сотруднике.

6. Реализовать класс монома от нескольких переменных, содержащий набор степеней, имён переменных и коэффициент. В классе должны быть определены и реализованы:
    - конструкторы (один из которых конструктор от char const *: например, “<5*x^2y^3z>”);
    - методы доступа к членам класса;
    - перегруженные операторы: +=, +, -=, -. *=, *, ==, !=;
    - перегруженные операторы выгрузки из потока / вставки в поток для корректного ввода / вывода монома.

   На основе реализованного класса монома необходимо реализовать класс полинома от многих переменных. Класс полинома представляет собой контейнер мономов, основанный на структуре данных вида двусвязный список. В классе полинома необходимо реализовать операции +=, +, -=, -, *=, *, ==, !=.

   Для демонстрации работы вашей программы реализуйте возможность обработки текстового файла следующего вида:

   <4x^2y^3-xy^2+4xy+1>+<x^2y^3+2xy^2+3x^2y^2-4xy+5>;
   <4x^2y^3 +4xy+1>*< x^2y^3+2xy^2+3x^2y^2-4xy+5>;
   < x^2y^3+2xy^2+3x^2y^2-4xy+5>-<4x^2y^3-xy^2+4xy+1>;
   < x^2y^3+2xy^2>==< x^2y^3+2xy^2>;
   < x^2y^3+2xy^2>!=< x^2y^3+2xy^2>;

   Замечание. Арифметические операции, возвращающие новый объект, необходимо реализовать с помощью соответствующих им операций, модифицирующих вызывающий объект: например, операция + должна быть реализована с помощью операции +=. Продемонстрировать передачу аргументов в функции по значению и по ссылке. Также необходимо продемонстрировать возврат объекта из функции. Реализуйте возможность сохранения и восстановления ваших объектов в/из потоков.

7. Реализуйте класс memory_cell, в котором содержится имя переменной, которое является строкой, её целочисленной значение и вспомогательные поля, которые Вам необходимы. Через аргументы командной строки в программу подается файл с инструкциями вида

```
myvar=15;
ccc=myvar/3;
{
        bg=25;
        ccc=bg+myvar;
        {
        param=0;
        param=ccc/bg+15;
        print param;
        }
        print ccc;
}
bg=ccc*myvar;
print;
```

Файл не содержит ошибок. Блоки кода выделяются с помощью фигурных скобок (‘{‘, ‘}’) и определяют область видимости переменных. Переменные, которые покидают область видимости, должны быть очищены. Реализуйте интерпретатор для обработки инструкций из входного файла с выполнением всех простых арифметических операций (+, -, *, /, %) и операции print (вывод в консоль либо текущего значения переменной (при указании имени переменной в инструкции), либо (при отсутствии имени переменной в инструкции) значений всех переменных с указанием их имён). При объявлении переменной необходимо выделить память под структуру переменной и сохранить информацию о переменной в структуре данных вида односвязный список. Имя переменной может иметь произвольную длину и содержать только символы латинских букв и символы цифр (при этом имя переменной не может начинаться с символа цифры). Заглавные и прописные буквы не отождествляются. В конце работы приложения необходимо очистить всю выделенную динамическую память. В случае использования в вычислениях не объявленной переменной или переменной, которая недоступна из текущей точки кода, необходимо вывести сообщение об ошибке и завершить работу интерпретатора.

8. На вход программе подается набор путей к текстовым файлов, которые содержат инструкции вида:

```
{
        ShowVar;
        var1=new(<size>);
        {
        var2=new(<size1>);
        ShowVar;
        {
                {
                        var3=new(<size2>);
                }
                var5=new(<size3>);
                ShowVar;
                        var21=new(<size12>);
                }
                var22=new(<size21>);

        }
        var2=new(<size1>);
        ShowVar;
}
```

Количество инструкций и их порядок произволен. Приложение должно поочерёдно обработать инструкции из каждого переданного файла. Оператор new запрашивает нужный размер оперативной памяти. Оператор ShowVar выводит на экран все видимые из данного места кода переменные, с указанием размеров памяти, который ими занят. Максимально доступный объем оперативной памяти определяется параметром N, являющимся аргументом командной строки. Фигурные скобки определяют блоки кода, которые определяют области видимости переменных. Если переменная покидает область видимости, она становится недоступной. При покидании области видимости, занятая память не возвращается автоматически. При этом учтите, что, если очередной вызов оператора new не может быть выполнен, то необходимо запустить операцию освобождения памяти под покинувшие свою область определения переменные, а затем вновь попытаться выделить требуемый блок. При невозможности выделения блока, необходимо завершить выполнение приложения.

9. Разработайте приложение для проведения лотереи (например, аналога Русского лото). Ваше приложение должно обеспечивать генерацию билетов для очередного тиража лотереи. Количество генерируемых билетов произвольно и может быть велико (> 20’000’000 шт.). Учтите ситуацию, что не все сгенерированные билеты могут участвовать в тираже (это типичная ситуация, которая возникает при неполной реализации билетов к тиражу). Смоделируйте проведение розыгрыша: на каждом ходе проверяйте, появился ли победитель; предусмотрите систему выигрышей; предоставьте возможность поиска билетов по заданным критериям: номеру билета, величине выигрыша, и т. д. Сохраняйте информацию о проведенных тиражах для обеспечения поиска данных в будущем. Реализуйте функционал обработки данных таким образом, чтобы тип коллекции/адаптера, в котором размещаются Ваши данные, являлся параметром шаблона. Продемонстрируйте обработку данных с использованием std::queue, std::map, а также собственных реализаций АВЛ-дерева и двусвязного списка.

10. Разработайте приложение для обработки данных логистической компании. Информация о доставке груза должна содержать информацию о грузе (название, вес, отправитель, получатель, стоимость, стоимость доставки, содержимое и т. д.), а также информацию об участках маршрута доставки (для участка необходимо хранить: пункт отправления, время отправления, пункт назначения, время доставки в пункт назначения, вид транспорта (автомобильный, железнодорожный, морской, авиационный)). Ваше приложение должно обеспечить возможность хранения и обработки данных о доставках (поиск по заданным критериям, добавление/удаление доставок). Для демонстрации работы реализуйте генератор (на базе паттерна “фабричный метод”), выдающий случайным образом сформированную информацию о доставке (необходимо проследить за тем, чтобы пункт назначения n −ного участка и пункт отправления $(n + 1)$ −го участка совпадали). Реализуйте функционал обработки данных таким образом, чтобы тип коллекции/адаптера, в котором размещаются Ваши данные, являлся параметром шаблона. Продемонстрируйте обработку данных с использованием std::deque, std::unordered_map, а также собственных реализаций красно-чёрного дерева и хеш-таблицы.

11. Разработайте приложение для проведения лотереи (например, аналога Спортлото 6 из 49 или 5 из 36). Ваше приложение должно обеспечивать генерацию билетов для очередного тиража лотереи. Количество генерируемых билетов произвольно и может быть велико (> 20’000’000 шт.). Учтите ситуацию, что не все сгенерированные билеты могут участвовать в тираже (это типичная ситуация, которая возникает при неполной реализации билетов к тиражу). Смоделируйте проведение розыгрыша: на каждом ходе проверяйте, появился ли победитель; предусмотрите систему выигрышей; предоставьте возможность поиска билетов по заданным критериям: номеру билета, величине выигрыша, и т. д. Сохраняйте информацию о проведенных тиражах для обеспечения поиска данных в будущем. Реализуйте функционал обработки данных таким образом, чтобы тип коллекции/адаптера, в котором размещаются Ваши данные, являлся параметром шаблона. Продемонстрируйте обработку данных с использованием std::vector, std::forward_list, а также собственных реализаций стека и очереди на базе односвязного списка.

12. На вход программе подаются файлы с данными диалогов пользователей некой социальной сети. Формат файла имеет следующий вид:

<user name> <time>: <message>

, при этом временная метка <time> имеет точность до миллисекунд. Поле <message> имеет произвольный размер. Обработайте поступающие файлы с использованием объекта типа std::set. В качестве данных храните указатели на сообщения пользователей с необходимыми атрибутами. В качестве набора ключевых полей, по которому будет выполняться сравнение, используйте <user name> и <time>. Приложение должно предоставить возможности для:

- вывода всех сообщений заданного пользователя;
- вывода сообщений заданного пользователя для заданного временного интервала;
- вывод всех сообщений из заданного временного интервала.

Также реализуйте опции удаления заданного сообщения и удаление сообщений заданного пользователя.

Приложение 1. Требования к оформлению пояснительной записки.

В основной части пояснительной записки необходимо для каждого реализованного задания описать алгоритм решения задания, использованные средства языка программирования C++, а также использованные при разработке алгоритмы и структуры данных.

Структура пояснительной записки:

- Титульный лист
- Содержание
- Введение
- Основная часть
- Вывод
- Список использованных источников
- Приложения

Оформление пояснительной записки:

- Поля: левое 20мм, остальные 15мм
- Нумерация страниц: начиная с титульного листа, индексация инкрементальная начиная с 1, по центру страницы; на титульном листе номер страницы не указывается
- Заголовки и подзаголовки разделов: шрифт Times New Roman 16pt, междустрочный интервал 1.5pt, выравнивание по левому краю
- Основной текст: шрифт Times New Roman 14pt, междустрочный интервал 1.15pt, выравнивание по ширине; абзацные отступы
- Рисунки: выравнивание по центру; под рисунком должна находиться подпись в формате

  Рисунок #. <Описание рисунка>

  , где # - номер рисунка при сквозной нумерации рисунков по всей пояснительной записке, индексация инкрементальная начиная с 1. Оформление подписи к рисунку: шрифт Times New Roman 12pt, курсивный, междустрочный интервал 1pt, выравнивание по центру; рисунок и подпись к нему должны находиться на одной странице
- Таблицы: Выравнивание по центру; над таблицей должна находиться подпись в формате

  Таблица #. <Описание таблицы>

  , где # - номер таблицы при сквозной нумерации таблиц по всей пояснительной записке, индексация инкрементальная начиная с 1. Оформление подписи к таблице: шрифт Times New Roman 12pt, курсивный, междустрочный интервал 1pt, выравнивание по левому краю; таблица и подпись к ней должны находиться на одной странице
- Листинги: Шрифт Consolas 12pt, междустрочный интервал 1pt, выравнивание по левому краю; над листингом должна находиться подпись в формате

  Листинг #. <Описание листинга>

  , где # - номер листинга при сквозной нумерации листингов по всей пояснительной записке, индексация инкрементальная начиная с 1. Оформление подписи к листингу: шрифт Times New Roman 12pt, курсивный, междустрочный интервал 1pt, выравнивание по левому краю; листинг и подпись к нему должны находиться на одной странице
- Список использованных источников: оформление по ГОСТ 7.0.100-2018