ОБРАЗОВАТЕЛЬНОЕ ЧАСТНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
ПРАВОСЛАВНЫЙ СВЯТО-ТИХОНОВСКИЙ ГУМАНИТАРНЫЙ УНИВЕРСИТЕТ
(ПСТГУ)

Факультет информатики и прикладной математики

Кафедра информатики

ФОНД
ОЦЕНОЧНЫХ СРЕДСТВ
ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ
ПО ДИСЦИПЛИНЕ

«Практикум по олимпиадному программированию (часть 2)»

02.03.03 «Математическое обеспечение и администрирование информационных систем»

Профиль:

Администрирование информационных систем

Квалификация выпускника бакалавр

Форма обучения: очная

Москва, 2020 г.

Год начала обучения по учебному плану: 2019

Фонд оценочных средств для текущего контроля успеваемости разработан на основе рабочей программы дисциплины «Практикум по олимпиадному программированию (часть 2)», входящей в состав образовательной программы 02.03.03 «Математическое обеспечение и администрирование информационных систем». Для текущего контроля успеваемости студентов в целях проверки процесса достижения результатов обучения и уровня компетенций, проводится 6 контрольных работ.

Контрольная работа №1

Базовые средства разработки для языка Java.

Цель работы:

Целью данной работы является формирование комфортного окружения для разработки программного обеспечения на языке Java.

Задачи:

1. Установка Java Development Kit в своей учетной записи.
2. Запуск и компиляция приложения HelloWorld.
3. Решение задач по вариантам.

Требования к сдаче

Программа должна быть загружена на сервер в соответствующую директорию и скомпилирована. Должен быть предоставлен отчет с исходным кодом программиы и кратким описанием.

Варианты:

1. реализовать сортировку методом пузырька;
2. реализовать сортировку выбором;
3. реализовать сортировку вставками;
4. реализовать сортировку Шелла;
5. реализовать сортировку слиянием;
6. реализовать быструю сортировку;
7. реализовать скалярное произведение двух векторов;
8. реализовать транспонирование матрицы;

9. реализовать суммирование двух матриц;
10. реализовать умножение матрицы на вектор;
11. реализовать произведение двух матриц;
12. реализовать вычисление матричной нормы для n=2;
13. реализовать вычисление матричной нормы при n равной бесконечности.

Контрольная работа №2

Разработка простейшего класса.

Требования к заданию

В проекте необходимо реализовать два публичных класса, один из которых, например Main, используется для инициализации экземпляров второго класса имеющего состояния и методы описанные для соответствующего варианта задачи. Другими словами один основной класс демонстрирует функционал в соответствии с вариантом задачи, а второй демонстрирует работоспособность основного класса. Каждый публичный класс должен размещаться в отдельном файле. Необходимо таким образом реализовать основной класс, чтобы из вспомогательного класса Main невозможно было напрямую обратиться к состояниям основного класса, а только через механизм сообщений. Проект должен быть выгружен и скомпилирован на сервере и предоставлен отчет о его выполнении.

Варианты:

1. Класс описывающий квадратную матрицу с операцией вычисления выполнения условия диагонального преобладания.
2. Квадратная матрица с операцией, определяющей, является ли указанный элемент одновременно наименьшим в своей строке и наибольшим в своём столбце.
3. Класс описывающий идеальный газ состоящий из n частиц и операцией вычисления средней кинетической энергии молекулы этого газа.
4. Класс описывающий сотрудников предприятия с полями: фамилия, имя, отчество, дата рождения, стаж работы и операцией вычисления сотрудника с самым
5. большим/маленьким стажем работы, самого молодого или самого старшего сотрудника.
6. Класс описывающий студентов с полями: фамилия, имя, отчество, номер группы с операцией поиска записи студента по строке, где поиск производится по его
7. фамилии, имени и отчеству.

8. Класс описывающий верхнюю треугольную матрицу и вектор правой части с операцией обратного хода метода Гаусса поиска решения системы уравнений.
9. Класс описывающий облако точек с операцией поиска расстояния между двумя произвольными точками.
10. Класс описывающий ненаправленный граф (дерево) с операцией поиска детей заданного узла.
11. Класс описывающий целочисленные матрицы с операцией сложения двух произвольных матриц.
12. Класс прямых на плоскости с операцией вычисления точки пересечения.
13. Класс векторов в трёхмерном пространстве с операцией векторного и скалярного произведения.
14. Класс треугольников в трёхмерном пространстве с операцией вычисления площади.
15. Класс кубов в трехмерном пространстве с операцией вычисления объема.

Контрольная работа №3

Реализовать модель вселенной. Каждый элемент вселенной должн быть объектом некоего публичного класса, который инициализируется вспомогательным публичным классом порождающим эту вселенную. При инициализации экземпляров класса элементов моделируемой вселенной необходимо подсчитывать количество элементов вселенной используя статичное экземплярное поле защищенное от изменения из объектов внешних классов путем реализации статичного метода.

Варианты:

1. Реализовать вычисление центра масс вселенной.
2. Реализовать вычисление среднего вектора направления движения частиц вселенной.
3. Реализовать вычисление общей массы вселенной и средней массы одной частицы при условии, что масса каждой частицы в общем случае разная.
4. Вычислить среднюю кинетическую энергию частиц вселенной.
5. Вычислить средний радиус вселенной.
6. Вычислить радиус-вектор центра вселенной.
7. Вычислить средний радиус вектор направления движения вселенной.

8. Вычислить среднюю силу притяжения действующую на каждую частицу вселенной.
9. Вычислить силу притяжения действующую на частицу массой M со стороны вселенной находящейся в заданной координате пространства.
10. Вычислить суммарную кинетическую энергию частиц вселенной.
11. Вычислить максимальное расстояние между двумя частицами вселенной.
12. Определить частицу вселенной, на которую действует максимальная сила со стороны соседних частиц.
13. Вычислить объем вселенной.

Контрольная работа №4

Данную работу необходимо реализовать используя знания о рекурсивном обходе элементов связного ациклического графа (дерева).

Задание заключается в реализации класса Tree представляющего из себя связный ациклический граф (дерево), элементы которого задаются из вспомогательного класса Main. При этом класс Tree должен содержать метод возвращающий детей getChildren для заданного узла и должна быть реализована возможность вызова данного метода, как из внешнего класса, так и из экземпляра класса Tree. Помимо этого класс Tree должен содержать метод рекурсивного обхода всего дерева getTree с применением метода getChildren. Дополнительно необходимо реализовать рекурсивный обход дерева из внешнего вспомогательного класса с применением реализованного метода getChildren.

Варианты:

					
1	2	3	4	5	6
					
7	8	9	10	11	12

Контрольная работа №5

Во время выполнения лабораторной работы требуется разработать на языке Java один из классов, перечисленных в таблице. В классе должен быть реализован интерфейс Comparable<T> и переопределён метод toString. В методе main вспомогательного класса Test нужно продемонстрировать работоспособность разработанного класса путём сортировки массива его экземпляров.

Варианты:

1. Класс последовательностей целых чисел с порядком на основе разности максимального и минимального числа.
2. Класс, представляющий множество материальных точек, заданных координатами в трёхмерном пространстве и массой, с порядком на основе расстояния от центра масс до начала координат.
3. Класс шаров, определяемых их координатами в трёхмерном пространстве и радиусами, с порядком на основе их объемов.
4. Класс кругов, заданных координатами их центров и радиусов, с порядком на основе их площадей.
5. Класс пар векторов в трёхмерном пространстве с порядком на основе длины их векторного произведения.
6. Класс треугольников, заданных координатами точек на плоскости, с порядком на основе площади треугольника.
7. Класс стеков целых чисел с порядком на основе максимального значения на стеке.
8. Класс вещественных векторов произвольной размерности с порядком на основе длины вектора.
9. Класс, представляющий множество вещественных векторов в n-мерном пространстве с порядком на основе длины суммы векторов множества.
10. Класс многоугольников с порядком на основе максимальной длины стороны многоугольника.
11. Класс точек в трехмерном пространстве с порядком на основе максимальной удаленности от центра системы координат.
12. Класс пар векторов в трёхмерном пространстве с порядком на основе значения их скалярного произведения.

Контрольная работа №6

Цель работы:

Изучение обобщённых итераторов и экземплярных вложенных классов языка Java.

Исходные данные

Интерфейсы Iterator и Iterable

Обобщённый интерфейс java.util.Iterator является контрактом, которому должны удовлетворять классы, объекты которых предназначены для перебора элементов некоторого множества значений:

```
public interface Iterator<E> {
    boolean hasNext ( ) ;
    E next ( ) ;
}
```

Объекты классов, реализующих этот интерфейс, называются итераторами. Тип перебираемых значений задаётся типовым параметром E, метод hasNext итератора возвращает true, если ещё остались не рассмотренные значения, а метод next возвращает следующее значение. Использование итератора можно проиллюстрировать следующим примером. Пусть в переменной it находится ссылка на итератор, перебирающий объекты некоторого класса SomeType. Тогда перебор всех объектов можно организовать в цикле такого вида:

```
while ( it.hasNext ( ) )  {
    SomeType x = it.next ( ) ;
    // Сделать что-то с объектом x
}
```

Контейнерные классы, как правило, реализуют обобщённый интерфейс Iterable, в котором объявлен метод iterator. Этот метод предназначен для создания нового итератора для перебора объектов, содержащихся в контейнере (т.е. в объекте контейнерного класса):

```
public interface Iterable<T> {
```

```
    Iterator <T> iterator ( );
}
```

Разрешено создавать сразу несколько итераторов для одного контейнера. Эти итераторы работают совершенно независимо, что позволяет, например, реализовать на двух итераторах перебор всех пар объектов, содержащихся в контейнере:

```
Iterator <SomeType> i = container.iterator ( );
while ( i.hasNext( ) ) {
    SomeType a = i.next ( ) ;
    Iterator <SomeType> j = container.iterator ( ) ;
    while ( j.hasNext( ) ) {
            SomeType b = j.next ( ) ;
            // Сделать что-то с парой ( a ,  b )
    }
}
```

Специальная форма оператора for

Контейнер, класс которого реализует интерфейс Iterable, можно использовать в специальной форме оператора for:

```
for  ( тип_элемента переменная  :  контейнер )    . . .
```

Эта форма оператора for является сокращённой записью следующего фрагмента кода:

```
Iterator <тип_элемента> it = контейнер . iterator( ) ;
while  ( it.hasNext( ) )  {
    тип_элемента переменная = it.next( );
    . . .
}
```

Тем самым, перебор всех пар объектов контейнера можно переписать как

```
for ( SomeType a  : container ) {
    for  ( SomeType b : container ) {
```

```
        // Сделать что то с парой ( a ,  b )
        }
}
```

Для общности специальную форму оператора for также разрешено использовать для перебора элементов массива.

Реализация итераторов через вложенные классы

Как правило, итератору необходим доступ к внутреннему состоянию контейнера. Чтобы не нарушать инкапсуляцию, удобно реализовать итератор в виде экземплярного вложенного класса внутри контейнерного класса. В качестве примера рассмотрим класс Su ffixList, пр Java представляются классом StringBuilder. Тем самым, объект класса Su ffixList будет контейнером для единственного объекта класса StringBuilder и будет предоставлять итератор по суффиксам строки, хранящейся в этом объекте.Внутри класса Su ffixList мы объявим вложенный экземплярный класс Su ffixIterator, в поле храниться индекс первого символа следующего суффикса. При создании итератора в поле pos будет записываться 0. Значение поля pos будет увеличиваться на единицу при каждом вызове метода next.

Листинг1: Test.java

```
public class Test {
      public static void main ( String [ ] args ) {
            StringBuilder b = new StringBuilder("qwerty");
            SuffixList suff = new SuffixList(b);
            for ( String s : suff ) System.out.println(s);
            b.insert(1,'x');
            for ( String s : suff ) System.out.println(s);
      }
}
```

Листинг2: SuffixList.java

```
import java.util.Iterator;
public class SuffixList implements Iterable<String> {
      private StringBuilder s;
```

```
    public SuffixList (StringBuilder s) { this.s = s; }
    public Iterator<String> iterator() { return new SuffixIterator( ) ; }
    private class SuffixIterator implements Iterator<String> {
        private int pos;
        public SuffixIterator () {
            pos = 0;
        }
        public boolean hasNext () {
            return pos < s.length( );
        }
        public String next () {
            return s.substring(pos++, s.length());
        }
    }
}
```

Критерии оценки контрольных работ:

Шкала оценки		**Критерии оценки**
Оценка	**Баллы**	
5 (отлично)	9 - 10	Обучающийся: • полно излагает изученный материал, • дает правильное определение понятий; • обнаруживает понимание материала, • может обосновать свои суждения, • может привести необходимые примеры не только из учебных пособий, но и самостоятельно составленные; • количество небольших замечаний не более 5; • реализовывает программный код без ошибок компиляции; • программа работает корректно и проходит все тесты;
4 (хорошо)	7 – 8	Обучающийся: • полно излагает изученный материал, • дает правильное определение понятий; • обнаруживает понимание материала, • может обосновать свои суждения, • может привести примеры; • количество ошибок не более 5; • реализовывает программный код без ошибок

		компиляции; • программа работает корректно и проходит все тесты;
3 (удовлетворительно)	5 - 6	Обучающийся: • обнаруживает знание и понимание основных положений; • но излагает материал неполно и допускает неточности в определении понятий или формулировках; • не умеет достаточно глубоко и доказательно обосновать свои суждения и привести свои примеры; • количество серьезных ошибок не более 5; • реализовывает программный код без ошибок компиляции; • программа работает корректно и проходит все тесты;
2 (неудовлетворительно)	0 - 4	Обучающийся: • обнаруживает незнание большей части соответствующего раздела изучаемого материала, • допускает ошибки в формулировке определений, искажающие их смысл; • количество серьезных ошибок более 5; • нет реализации программного кода либо программа работает не корректно.

Максимальное число баллов, которое может получить студент за работу в течение семестра, равно 80-ти.

Авторы: доц. к.ф.-м..н. Посевин Д.П.

Программа одобрена на заседании кафедры Информатики от «29» мая 2020 года, протокол № 05-20.