УТВЕРЖДЕНО
Проректор по учебной работе
и довузовской подготовке
А. А. Воронов
9 января 2018 г.

ПРОГРАММА

по дисциплине: **Информатика. Продвинутый курс**
по направлению подготовки: 03.03.01 «Прикладные математика и физика»
физтех-школа: **ФАКТ**
факультет: **ФАКИ**
кафедра: **информатики и вычислительной математики**
курс: 1
семестр: 2

Трудоёмкость:

вариативная часть – 3 зачет. ед.	Экзамен – нет
лекции – 30 часов	Зачет дифф. – 2 семестр
практические (семинарские)	Задания – 2
занятия – нет	Контрольные работы – 2
лабораторные занятия – 60 часов	Самостоятельная работа – 90 часов

ВСЕГО АУДИТОРНЫХ ЧАСОВ – 90

Программу составили: чл.-корр. РАН, д.ф.-м.н. И. Б. Петров
доцент, к.т.н. В. К. Хохлов
доцент, к.ф.-м.н. К. А. Беклемышева

Программа принята на заседании
кафедры информатики и вычислительной математики
14 ноября 2017 г.

Заведующий кафедрой
чл.-корр. РАН И. Б. Петров

Структура преподавания дисциплины

Общее в C++ и C. Дополнительные необъектно-ориентированные возможности C++. Ссылки. Лямбда-выражения. Перегрузка функций. Обработка исключений. Пространства имен.

Механизмы абстракции. Синтаксис классов изолированных (невзаимодействующих) объектов в языке C++. Классы, представляющие нормативное знание. Конструкторы и деструкторы. Интерфейс и его реализация.

Трансформирование базовых типов языка C++ в классы – основной источник формирования классов в языке C++. Классы-оболочки (Wrappers) с минимальным интерфейсом для базовых типов данных (примитивов). Полиморфизм конструкторов. Методы доступа. Ключевое слово this. Постоянные и модифицируемые члены класса. Массивы объектов. Классы-оболочки с полным интерфейсом инкапсулированного базового типа. Перегрузка операторов в C++. Инкапсуляция массивов базовых типов: абстрактные типы данных. Методы доступа. Перегрузка оператора индекса.

Дружественные функции: функции, дружественные классам, – функции за пределами классов (нарушение концепции объектно-ориентированного программирования). Инкапсуляция дружественных функций в класс Visitor. Объектно-ориентированное программирование без использования дружественных функций: классы данных и классы функций. Потоки ввода–вывода данных в C++. Форматируемый ввод/вывод и манипуляторы ввода/вывода. Перегрузка пользовательских операторов ввода/вывода как дружественных функций. Файловый ввод/вывод. Неформатируемый двоичный ввод/вывод (байтовые потоки). Инкапсуляция объектов и вертикальное делегирование функций.

Адресное пространство приложения. Динамические члены класса. Динамическая память (*куча*) и статическая память (*стек*). Операции с динамической памятью (кучей). Конструкторы и деструкторы объектов в динамической памяти. Проблема утечки памяти в C++. Статические переменные – члены класса. Инициализация статических членов класса.

Копирование объектов. Присваивание объектов. Передача и возвращение функциями объектов по значению. Конструкторы копий. Клонирование объектов в C++. Передача и возвращение ссылок на объекты.

Инкапсуляция массивов объектов. Классы-оболочки контейнеров с расширенным интерфейсом. Специальные методы создания объектов. Создание объектов посредством статических методов. Класс Singleton. Косвенное создание объектов и горизонтальное делегирование функций. Класс-заместитель Proxy. Классы-оболочки для указателей базовых типов. Перегрузка операций указывания и разыменования. Интеллектуальные указатели.

Повторное использование классов. Открытое наследование классов. Распространение и перегрузка наследуемых методов. Вызов конструкторов базового класса (суперкласса). Наследование функций и операций (Inheriting

operations and functions). Перегрузка оператора присвоения. Множественное и виртуальное наследование. Закрытое наследование классов. Классы-адаптеры. Классы-композиты, в объекты которых вложены объекты других классов. Делегирование функций.

Шаблоны. Полиморфизм функций и параметризованные функции (шаблоны). Параметризованные классы (шаблоны). Статический полиморфизм. Шаблоны интеллектуальных указателей.

Шаблоны классов-контейнеров. Вектор (массив). Строка. Связанный список. Обработка исключительных ситуаций.

Внутренние и дружественные классы. Реализация двумерного вектора. Перегрузка оператора двойного индекса. Шаблоны итераторов для классов-контейнеров.

Стандартная библиотека шаблонов STL (Standard Template Library). Классы-контейнеры и итераторы. Обобщенные алгоритмы. Функторы.

Классы с виртуальными функциями. Подстановочный критерий Барбары Лисков. Структура объектов и таблицы виртуальных функций. Динамический полиморфизм.

Чистые виртуальные функции и абстрактные классы. Полиморфное наследование абстрактных классов в C++. Производящие функции и фабрики объектов.

Динамическое приведение типов и идентификация (RTTI).

Наследование классов с расширением интерфейса. Шаблоны оболочек-адаптеров и внешний полиморфизм.

Одновременное (параллельное) выполнение потоков вычислений.

Задание 1

(*срок сдачи 26–31 марта*)

Списки

Реализовать класс List для создания структуры данных двухсвязного списка, каждый элемент которого Node хранит уникальный, случайно сгенерированный ключ.

Класс должен содержать методы:

- добавление элемента в начало и конец списка (*аргументы*: значение ключа для нового узла);
- вывод списка на экран;
- поиск элемента по значению ключа (*аргументы*: значение ключа);
- удаление элемента из начала списка, из конца списка, а также по значению ключа (*аргументы*: значение ключа);
- поиск количества элементов.

Программа должна создать заданный пользователем список и продемонстрировать работоспособность всех методов класса.

Бинарные деревья

Реализовать класс Tree для создания структуры данных бинарного дерева, каждый элемент которого Node хранит уникальный, случайно сгенерированный ключ.

Класс должен содержать методы:

- добавление узла (*аргументы*: значение ключа для нового узла);
- вывод дерева на экран;
- поиск узла дерева по значению ключа (*аргументы*: значение ключа);
- удаление узла по значению ключа (*аргументы*: значение ключа);
- поиск максимального элемента дерева;
- поиск суммы всех элементов дерева;
- поиск количества узлов дерева;
- поиск максимальной глубины дерева.

Программа должна создать дерево заданного размера и продемонстрировать работоспособность всех методов класса.

Векторы и матрицы

Реализовать класс Vector3 для трехмерных векторов. Перегрузить для него основные арифметические операции, в том числе:

- умножение на константу;
- сложение;
- вычитание;
- скалярное и векторное умножение.

Реализовать класс Matrix3 для матриц 3×3. Перегрузить для него основные арифметические операции:

- сложение;
- вычитание;
- определитель;
- умножение на вектор;
- умножение двух матриц.

Аналитическая геометрия

С использованием класса Vector реализовать класс Line, Plane, Segment и Triangle для прямой, плоскости, отрезка и треугольника соответственно. Реализовать класс CollisionDetector, который умеет находить пересечение любых двух объектов указанных типов (отрезок с отрезком, отрезок с плоскостью, треугольник с плоскостью и т.д.).

Задание 2

(*срок сдачи 23–28 апреля*)

Геометрические объекты

Реализовать класс Figure, содержащий общую информацию о геометрическом объекте и основные методы работы с ним:

- положение в пространстве;
- поворот;
- перемещение;
- растяжение вдоль осей.

Реализовать набор классов, наследованных от Figure (и, возможно, друг от друга): Triangle (треугольник), Sphere (сфера), Ellipsoid (эллипсоид), Cube (куб), Parallelepiped (параллелепипед), …

Реализовать в классе Figure статический метод CollisionDetector, который умеет определять, пересекаются ли два геометрических объекта.

Реализовать в классе Figure счетчик созданных объектов.

«Дождь»

В замкнутом кубическом объеме на верхней границе случайно генерируются различные объекты типа Figure с произвольной начальной скоростью, направленной вниз. После пересечения нижней границы объекты исчезают. Одновременно в объеме должно быть порядка 1000 объектов.

Программа должна работать без утечек памяти, проверить это можно при помощи valgrind.

Дополнительное задание. Визуализация процесса. Максимально увеличить количество объектов, которые могут одновременно находиться в объеме.

Бинарные деревья поиска

На основе класса Tree из первого задания реализовать класс SearchTree для создания структуры данных бинарного дерева поиска.

Универсальный список

На основе класса List из первого задания реализовать аналог list из STL.

Моделирование

Дополнительное задание ко всем задачам данного раздела – визуализация процесса, описанного в задаче.

«Задача трех тел»

В неограниченном трехмерном пространстве находится *N* шарообразных объектов различной массы и различного радиуса. В начальный момент времени они имеют случайные (либо заданные пользователем) векторы скоростей. Объекты взаимодействуют друг с другом согласно закону всемирного тяготения. Определить, какие из объектов столкнутся, и время столкновения.

Дополнительное задание. Вводится поправка к закону всемирного тяготения, которая обеспечивает диссипацию энергии – например, если спутник находился на стабильной орбите вокруг планеты, после введения поправки он будет на нее постепенно падать. Необходимо разработать модель этой поправки (например, приливные силы или гравитационные волны) и ввести ее в программу.

«Тренировка футболистов»

Команда футболистов отрабатывает индивидуальные действия. Каждый играет за себя. Цель футболиста – завладеть мячом и забить гол. Таким образом, он может выполнять два действия:

а) бороться за мяч;

б) бить по воротам, как только мяч оказался у него.

После удара по воротам вратарь выбивает мяч в произвольное место поля. Футболист забивает гол с определенной вероятностью, зависящей от расстояния до ворот. Команда играет определенное время, потом определяется победитель – по забитым мячам. Вывести на экран счет и последовательность действий каждого футболиста.

«Ветеран Броуновского движения»

В кубическом замкнутом объеме находится N «маленьких» шарообразных частиц массы m и одна «большая» массы M. В начальный момент времени они имеют случайные векторы скоростей. Частицы упруго отталкиваются друг от друга и от стенок. Построить график зависимости скорости «большой» частицы от времени.

«Воробьи»

Бабушка на скамейке периодически (пусть раз в Q минут) бросает воробьям по одной ровно N хлебных крошек. Когда бросают крошку хлеба, первый подлетевший к ней воробей хватает крошку, относит ее в сторону и съедает. За счет этого он пропускает «розыгрыш» следующей крошки. Первоначально у скамейки находится M воробьев. Каждую минуту к стае прилетает еще один воробей. Съев P крошек, воробей наедается и улетает. Чтобы количество воробьев не увеличивалось, должно выполняться соотношение $Q = N/P$.

Итак, воробей может выполнять действия:

- прилететь;
- ждать раздачи крошек;
- драться за крошку;
- отлететь с крошкой в сторону и съесть ее;
- улететь;

Вывести на экран последовательность действий каждого из воробьев.

Литература

Основная

1. *Страуструп Б.* Язык программирования C++: спец. изд. – М.: БИНОМ, 2008.
2. *Александреску А.* Современное проектирование на C++. Обобщенное программирование и прикладные шаблоны проектирования. – М.: ИД «Вильямс», 2008.

Дополнительная

1. *Буч Г.* Объектно-ориентированный анализ и проектирование. – 2-е изд. – М.: БИНОМ, 1998.
2. *Майерс С.* Эффективное использование STL. – СПб.: ПИТЕР, 2002.
3. *Влиссидес Дж.* Применение паттернов проектирования. Дополнительные штрихи. – М.: ИД «Вильямс», 2003.
4. *Прата С.* Язык программирования C++. Лекции и упражнения. – СПб.: ООО «ДиаСофтЮП», 2005.
5. *Топп У., Форд У.* Структуры данных в C++. – М.: ИД «Вильямс», 2000.
6. *Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж.* Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб.: ПИТЕР, 2003.
7. *Шилдт Г.* C++: базовый курс. – 3-е изд. – М.: ИД «Вильямс», 2016.

Электронные ресурсы, включая доступ к базам данных

1. http://judge.mipt.ru
2. http://cs.mipt.ru
3. http://acm.mipt.ru