

УТВЕРЖДЕНО  
Проректор по учебной работе  
и довузовской подготовке  
А. А. Воронов  
27 июня 2017 г.

## ПРОГРАММА

по дисциплине: **ИНФОРМАТИКА. Продвинутый курс**  
по направлению подготовки: **03.03.01 «Прикладная математика и физика»**  
физтех-школа: **ФЭФМ**  
факультет: **ФФКЭ**  
кафедра: **информатики и вычислительной математики**  
курс: 1  
семестр: 1

Трудоемкость:

Базовая часть – 3 зачет. ед.:

лекции: – нет

практические (семинарские)

занятия: – нет

лабораторные занятия: – 60 часов

Экзамен – нет

Зачет дифф. – 1 семестр

Две контрольные работы

Самостоятельная работа – 75 часов

ВСЕГО АУДИТОРНЫХ ЧАСОВ – 60

Программу составили:

академик РАН В. П. Иванников,  
доцент, к.ф.-м.н. П. Н. Коротин  
доцент, к.т.н. В. К. Хохлов

Программа принята на заседании

кафедры информатики и вычислительной математики

16 июня 2017 г.

Заведующий кафедрой,  
чл.-корр. РАН

И. Б. Петров

Структура преподавания дисциплины.

**Введение в теорию алгоритмов.** Интуитивное понятие алгоритма. Свойства алгоритмов. Понятие об исполнителе алгоритма. Алгоритм как преобразование слов из заданного алфавита. Связь понятия алгоритма с понятием функции. Машина Тьюринга. Нормальные алгорифмы Маркова. Вычислимые функции и их свойства. Невычислимые функции. Различные эквивалентные определения множества вычислимых функций. Алгоритмическая сложность.

**Алгоритмические языки.** Характеристика алгоритмических языков и их исполнителей. Понятие трансляции.

Понятие о формальных языках. Способы строгого описания формальных языков, понятие о метаязыках. Алфавит, синтаксис и семантика алгоритмического языка. Описание синтаксиса языка с помощью металингвистических формул и синтаксических диаграмм.

Языки программирования. Общие характеристики языков программирования. Алфавит, имена, служебные слова, стандартные имена, числа, текстовые константы, разделители. Препроцессор и комментарии.

Типы данных, их классификация. Переменные и константы. Скалярные типы данных и операции над ними. Старшинство операций, стандартные функции. Выражения и правила их вычисления. Оператор присваивания.

Файлы. Стандартные функции ввода-вывода.

Простые и сложные операторы. Пустой, составной, условный операторы. Оператор варианта. Оператор перехода.

Оператор цикла. Программирование рекуррентных соотношений.

Составные типы данных. Массивы.

Описание функций (процедур). Формальные и фактические параметры. Способы передачи параметров. Локализация имен. Побочные эффекты. Итерации и рекурсии.

Ссылочный тип данных. Методы выделения памяти: статический, динамический и автоматический. Структуры. Битовые поля. Объединения. Перечисления. Декларация typedef.

**Алгоритмы сортировки.** Понятие внутренней и внешней сортировки. Устойчивая сортировка. Сортировка in-place. Сортировка простыми вставками, простым выбором, метод «пузырька». Шейкер сортировка. Метод Шелла. Быстрая сортировка Хора. Сортировка слиянием. Пирамидальная сортировка. Оценка трудоемкости.

**Алгоритмы и структуры данных.** Абстрактные структуры данных: список, стек, очередь, очередь с приоритетом, ассоциативный массив. Отображение абстрактных структур данных на структуры хранения: массивы, линейные списки, деревья.

Различные реализации ассоциативного массива: двоичные деревья поиска (AVL-деревья, красно-чёрные деревья), перемешанные таблицы (с

прямой и открытой адресацией, использование техники двойного хэширования при открытой адресации). Оценки алгоритмической сложности операций поиска, добавления и удаления элемента.

Классические алгоритмы: перебор с возвратом, жадные алгоритмы. Примеры алгоритмов работы с графами: поиск минимального остового дерева, поиск кратчайшего пути, задача коммивояжера.

### **Учебно-методическое и информационное обеспечение дисциплины** **Основная литература**

1. *Ворожцов А. В., Винокуров Н. А.* Практика и теория программирования. – М.: Физматкнига, 2008.
2. *Керниган Б. У., Ритчи Д. М.* Язык программирования С. – 2-е издание. – М.: Издательский дом «Вильямс», 2006.

### **Дополнительная литература**

1. *Кормен Т. Х., Лейзерсон Ч. И., Ривест Р. Л., Штайн К.* Алгоритмы: построение и анализ. – 3-е изд. – М.: Издательский дом «Вильямс», 2013.
2. *Дасгупта С., Пападимитриу Х., Вазирани У.* Алгоритмы. Пер. с англ. под ред. А. Шеня. – М.: МЦНМО, 2014.
3. *Шилдт Г.* Полный справочник по С. – М.: Издательский дом «Вильямс», 2005.
4. *Ахо А., Хопкрофт Дж., Ульман Дж.* Структуры данных и алгоритмы. – М.: Издательский дом «Вильямс», 2000.
5. *Вирт Н.* Алгоритмы и структуры данных. – СПб.: Невский Диалект, 2005.
6. *Седжвик Р.* Фундаментальные алгоритмы на С. – СПб.: ООО «ДиаСофтЮП», 2003.
7. *Митницкий В. Я.* Элементы теории алгоритмов и язык программирования С. – М.: МФТИ, 2001.

### **Пособия и методические указания**

1. *Прут В. В.* Функции алгебры логики. Алгоритмы на языке С. Методические указания. – М.: МФТИ, 2013.
2. *Прут В. В.* Полные системы функций алгебры логики. Алгоритмы на языке С. Методические указания. – М.: МФТИ, 2013.
3. *Прут В. В.* Основные комбинаторные алгоритмы на языке С. Методические указания. – М.: МФТИ, 2013.
4. *Прут В. В.* Алгоритмы обработки структур данных на языке С. Списки, стеки, деревья. Методические указания. – М.: МФТИ, 2013.
5. *Прут В. В.* Сортировка и поиск. Алгоритмы на языке С. Учебно-методическое пособие. – М.: МФТИ, 2014.
6. *Прут В. В.* Введение в графы. Алгоритмы на языке С. Учебно-методическое пособие. – М.: МФТИ, 2014.

7. Прут В. В. Свойства графов. Алгоритмы на языке С. Учебно-методическое пособие. – М.: МФТИ, 2014.

### Электронные ресурсы

1. <http://cs.mipt.ru>
2. <http://acm.mipt.ru>

## ЗАДАНИЕ 1

(срок сдачи 23–28 октября)

### 1. Машины Тьюринга

Обозначим как  $N_0$  множество всех неотрицательных целых чисел.

Описать машины Тьюринга, которые реализуют:

- 1.1. Счетчик четности. Выход машины Тьюринга равен 0 или 1 в зависимости от того, четно или нечетно число единиц в последовательности из 0 и 1, записанной на ленте машины Тьюринга. В конце последовательности стоит символ  $B$ . В начальном состоянии головка видит первый левый символ.
- 1.2. Инверсию заданного слова в алфавите  $\{0, 1\}$  (0 заменяет на 1, а 1 – на 0).
- 1.3. «Переворачивание» заданного слова в алфавите  $\{a, b, c\}$ .
- 1.4. Сложение двух чисел из множества  $N_0$ , записанных на ленте в виде последовательности единиц, а именно:

$$0 \rightarrow 0, 1 \rightarrow 01, 2 \rightarrow 011, 3 \rightarrow 0111, 4 \rightarrow 01111, \dots$$

Назовём эту запись *единичной записью* числа. Записи чисел разделены на ленте несколькими пустыми ячейками. Рассмотрите различные варианты начального положения головки машины Тьюринга.

- 1.5. Сложение двух чисел из  $N_0$ , данных в двоичной системе счисления.
- 1.6. Распознавание правильных скобочных выражений. Правильное скобочное выражение – это слово в алфавите  $A = \{(\,)\}$ , которое может получиться, если из арифметического выражения удалить все символы, кроме скобок. Примеры правильных скобочных выражений: пустое слово,  $()$ ,  $((\ ))$ ,  $(\ )(\ )$ . Примеры неправильных скобочных выражений:  $)$ ,  $(\ )(\ )$ ,  $(\ )))$ ,  $((\ )$ . Результат работы: слово «YES», если скобочное выражение правильное, и слово «NO» – иначе.
- 1.7. Перемножение двух чисел из  $N_0$ , заданных в виде единичных записей. Числа записаны на ленте подряд.
- 1.8. Вычисление квадрата числа, заданного в виде единичной записи.

### 2. Алгоритмы Маркова

- 2.1. Записать нормальные алгоритмы Маркова, которые реализуют:
  - 2.1.1. Приписывание буквы  $X$  к входному слову справа.
  - 2.1.2. Задание 1.2.
  - 2.1.3. Задание 1.3.
  - 2.1.4. Задание 1.5.
  - 2.1.5. Задание 1.6.

2.1.6. Удвоение числа, заданного а) в виде единичной записи, б) в двоичной системе счисления.

2.2. В алфавите  $A = \{a, b\}$  описать нормальный алгоритм, который выдает в качестве результата пустое слово, если буквы  $a$  и  $b$  входят во входное слово в равном количестве, и любое непустое слово – иначе. В алгоритме должно быть не более четырех правил подстановки. Докажите правильность придуманного алгоритма.

2.3. Существует ли алгоритм Маркова, применимый только к двоичным записям простых чисел?

2.4. Верно ли, что человек для любого алгоритма Маркова может написать машину Тьюринга, реализующую ту же функцию (то есть множество слов, к которым они применимы, совпадает и для любого элемента из этого множества результаты работы алгоритма Маркова и машины Тьюринга совпадают)? Разрешима ли эта задача алгоритмически?

2.5. Верно ли, что человек для любого алгоритма Маркова и заданного входного слова может определить, применим алгоритм Маркова к этому слову или нет? Можно ли поручить эту программу компьютеру (в принципе, не принимая во внимание доступное время и вычислительные мощности)? Ответьте на те же вопросы при условии, что максимальное число правил ограничено числом 10.

### 3. Решение простых алгоритмических задач

В каждой задаче ответьте на вопрос о том, как растет время работы программы и используемая программой память с ростом параметра размера входных данных (например, параметра  $n$ ).

3.1. «Мах». Написать программу, которая выводит максимальное число из  $n$  заданных чисел. В первой строчке входа дано число  $n$ , а в следующей строчке указано  $n$  целых чисел.

3.2. «Числа Фибоначчи I». Написать программу, которая по данному  $n$  находит  $n$ -е число Фибоначчи  $F_n$ . Числа Фибоначчи определяются соотношениями

$$F_n = F_{n-1} + F_{n-2}, F_1 = F_2 = 1.$$

3.3. «Числа Фибоначчи II». Решить предыдущую задачу, используя идею рекурсии. Оценить число элементарных операций, которое необходимо сделать в рекурсивном и нерекурсивном алгоритмах вычисления числа  $F_n$ .

3.4. «Биномиальные коэффициенты». Написать программу, которая для данного натурального числа  $n$  находит коэффициенты в разложении

$$(1+x)^n = C_n^0 x^0 + C_n^1 x^1 + \dots + C_n^n x^n.$$

Использовать соотношения

$$C_n^n = C_n^0 = 1, C_n^k = C_{n-1}^k + C_{n-1}^{k-1}.$$

Оценить, как растет время работы вашей программы с ростом  $n$ .

3.5. «Простые числа». Написать программу, которая определяет, является ли введенное число  $n$  простым.

3.6. Написать программу, вычисляющую площадь односвязной прямоугольной фигуры, заданной перечислением пар целочисленных координат её вершин в произвольном порядке.

3.7. «Задача Иосифа».  $N$  человек, имеющие номера от 1 до  $N$  и расположившиеся по кругу друг за другом, считаются считалкой, состоящей из  $M$  слов. Расчет начинается с номера 1. Человек, на котором считалочка заканчивается – выбывает. А расчет (этой же считалочкой) продолжается с участника, следующего за выбывшим. Написать программу, которая для заданных  $N$  и  $M$  сообщает (в порядке выбывания) номера трех игроков, выбывших последними.

3.8. «Уравнение». Написать программу, которая в указанном интервале находит нетривиальный корень уравнения  $\operatorname{tg} x = x$  с погрешностью  $10^{-10}$ . Сколько итераций необходимо сделать, чтобы достичь указанной точности методами деления пополам, Ньютона, простых итераций?

#### 4. Жадные алгоритмы

4.1. «Атлеты». Написать программу, которая находит «башню» из атлетов максимальной высоты. Атлеты характеризуются двумя параметрами – массой и силой. Сила равна максимальной массе, которую атлет может держать на плечах. Известно, что если атлет тяжелее, то он точно сильнее. Подсказка: упорядочьте атлетов по силе и стройте башню сверху. Вверх естественно поместить самого слабого. Входом является число атлетов  $n$  и  $n$  пар (масса, сила).

4.2. «Отрезки». Написать программу, которая для множества заданных отрезков находит минимальное подмножество отрезков, объединение которых покрывает отрезок  $S = [0, 10000]$ . Число отрезков и координаты их концов заданы на входе. Все координаты целочисленные. Подсказка: покрывайте отрезок  $S$  пошагово, двигаясь слева направо. На каждом шаге будет непокрытая часть  $[x, 10000]$ . Из оставшихся отрезков выбирайте тот, который урежет непокрытую часть до  $[y, 10000]$ , где  $y$  максимальное. Решите эту задачу за время  $O(n \log n)$ , где  $n$  – количество данных отрезков.

4.3. «Коммивояжер». Написать программу, которая, используя один из возможных жадных алгоритмов, находит на плоскости ломаную линию, идущую из  $A$  в  $B$  по всем заданным точкам  $\{A_1, A_2, A_3, \dots, A_n\}$ , как можно меньшей длины. Есть ли такие входные данные, когда написанная программа находит не самый короткий путь? Если есть, приведите пример.

## ЗАДАНИЕ 2

(срок сдачи 4–9 декабря)

### 1. Структуры данных: списки, стек, очередь, деревья поиска и перемешанные таблицы (хэш-таблицы).

1.1. «Список I». Реализовать односвязный список, элементы которого содержат целые числа. Реализовать при этом функции *list\_new()* (создать новый список), *list\_delete(l)* (удалить список *l* и все его элементы), *insert(l, a)* (добавить элемент с заданным целым числом *a* в начало списка *l*), *remove(l, a)* (удалить из списка *l* все элементы, содержащие заданное целое число *a*), *print(l)* (вывести значения, хранящиеся в элементах списка *l*). Осуществите массовое и многоплановое тестирование всех реализованных функций.

1.2. Для структуры данных из задачи 1.1 реализовать функцию *first\_integers(N)* от *N*, которая конструирует список вида



(при  $N = 0$  список пустой) и возвращает как свое значение ссылку на этот список.

1.3. «Список II». Реализовать двусвязный список, элементы которого содержат целые числа. Реализовать функции: *list\_new()* (создать новый пустой список), *list\_delete(l)* (удалить список и все его элементы), *push(l, a)* (добавить новый элемент *a* в конец списка), *pop(l, x)* (извлечь последний элемент списка), *unshift(l, a)* (добавить новый элемент *a* в начало) и *shift(l, x)* (извлечь первый элемент списка). Последние пять функций в качестве первого аргумента получают указатель на список, а возвращают 1 или 0 в зависимости от того, успешно ли выполнена операция. Функции *push* и *unshift* во втором аргументе получают добавляемый элемент. Функции *pop* и *shift* во втором аргументе *x* получают адрес, куда следует поместить извлекаемый элемент. Реализуйте также функцию *reverse*, которая инвертирует список, ссылку на который получает в качестве аргумента.

1.4. «Скобки». Дано слово, состоящее из круглых и фигурных скобок. Написать программу, которая определяет, является ли введенное слово правильным скобочным выражением. Подсказка: постепенно считывайте скобки и используйте стек (можно использовать список из задачи 1.3 с командами *push* и *pop*) для хранения открывающих скобок, для которых пока не считаны парные закрывающие скобки. (Рекурсивное определение правильного скобочного выражения: слово называется правильным скобочным выражением, если все скобки в нем можно разбить на пары, так что в каждой паре скобка, стоящая ближе к левому концу слова, открывающая, а

вторая – закрывающая, при этом они имеют один и тот же тип, и, кроме того, слово, стоящее между парными скобками, является правильным скобочным выражением. Например, слова  $()$ ,  $\{()\}$ ,  $\{\{()\}\}$  – правильные скобочные выражения, а слова  $\{\{, \{\}\}$ ,  $\{\{()\}$  – неправильные скобочные выражения.)

1.5. «Калькулятор». Написать программу, которая, используя стек (используйте код, полученный при решении задачи 1.3), вычисляет значение арифметического выражения, заданного в постфиксной форме.

1.6. Написать программу, которая получает на вход арифметическое выражение в инфиксной форме и выводит это выражение в постфиксной форме. Программа должна работать за время, ограниченное линейной функцией от размера входного слова.

1.7. «Лабиринт». Написать программу, которая находит кратчайший путь из левого верхнего угла лабиринта в нижний правый угол либо определяет, что такого пути нет. Лабиринт задан в виде прямоугольного клеточного поля, белые клетки в котором считаются проходимыми, а черные – непроходимыми. В первой строчке входа заданы размеры лабиринта  $N$  и  $M$  по горизонтали и вертикали соответственно. Далее идут  $N$  строчек, в каждой из которых дано слово в алфавите  $\{\#, .\}$  из  $M$  символов. Символ '#' означает черную клетку, а '.' – белую. Выход программы должен содержать последовательность пар координат клеток, по которым нужно идти, либо слово «NO». Использовать очередь (например, список из задачи 1.3 с командами *push* и *shift*) для того, чтобы хранить новые найденные клетки. Последовательно из начала очереди брать (команда *shift*) клетки, чтобы проверить, можно ли из них сделать шаг в новые, не найденные пока клетки, которые помещать (команда *push*) в конец очереди. Как растет время работы алгоритма в зависимости от  $N$  и  $M$  в худшем случае?

1.8. «Двоичное дерево поиска». Реализовать структуру данных «двоичное дерево поиска» с функциями создания и удаления дерева, добавления пары (ключ, значение) и удаления пары по ключу, где ключ есть целое число, а значение – действительное число. Написать функции  $wfs(t)$  и  $dfs(t)$  обхода дерева в ширину и в глубину. В первом случае следует использовать очередь, а во втором – рекурсию или стек.

1.9. Для структуры данных «двоичное дерево поиска» реализовать функцию префиксного обхода дерева  $traverse(tree, f)$ , которая ко всем значениям, хранящимся в дереве  $tree$ , применяет функцию  $f(item, depth)$ . В качестве аргументов функция  $f$  получает ссылку на элемент дерева  $item$  и глубину  $depth$  этого элемента в дереве. Реализовать с помощью функции  $traverse$

вывод описания дерева в стандартный поток вывода (префиксное описание дерева).

1.10. «Записная книжка I». Написать программу, которая реализует функциональность телефонной записной книжки. А именно, из стандартного входа программа получает последовательность команд на добавление (*INSERT*) или поиск (*FIND*) записей. Примеры команд: *INSERT Sidorov 1234567*, *INSERT Ivanov 7654321*, *FIND Sidorov*. При выполнении команды *INSERT* программа добавляет пару (фамилия, номер) в своё хранилище и выводит строку «OK», если в хранилище нет записи с такой фамилией, или изменяет соответствующую указанной фамилии запись и выводит строку «*Changed. Old value = X*», если запись с такой фамилией уже есть в хранилище и соответствующий телефонный номер был *X*. При выполнении команды *FIND* программа выводит телефонный номер для указанной фамилии или выводит «NO», если указанной фамилии нет в справочнике. Следует использовать структуры, состоящие из двух элементов *name* и *number*. Использовать технику динамического выделения памяти для хранения записей. Оценить, как в среднем растёт число элементарных операций при выполнении команд *INSERT* и *FIND* с ростом числа хранимых записей. Записи хранить в массиве или списке. Рассмотреть два случая: а) записи хранятся в отсортированном по фамилиям (в алфавитном порядке) виде; в случае хранения в массиве записей используйте метод деления пополам (см. 3.8); б) записи хранятся в произвольном порядке (например, в порядке добавления).

1.11. «Записная книжка II». Решите задачу 1.10, используя двоичное дерево поиска.

1.12. «Записная книжка III». Решите задачу 1.10, используя AVL-дерево.

1.13. Приведите описание рекурсивной процедуры *check\_tree(h, p)*, которая проверяет, является ли дерево AVL-деревом.

1.14. «Записная книжка IV». Решите задачу 1.10, используя хэш-таблицу с разрешением коллизий методом цепочек либо методом открытой адресации.

1.15. «Поиск путей с заданной суммой». Написать программу, выводющую на экран «YES», если в данном дереве существует путь от корня до листа, сумма элементов в вершинах которого равна заданному целому числу *S*, и «NO» – в противном случае. Считается, что в пустом дереве сумма элементов пути равна 0.

1.16. Поставить численный эксперимент, чтобы определить, как зависит суммарное число столкновений и суммарное время работы при добавлении в хэш-таблицу с открытой адресацией и двойным хешированием *K* случай-

ных ключей при размере таблицы  $N$ . Нарисовать графики зависимости суммарного числа столкновений и суммарного времени работы при добавлении  $K$  ключей от степени заполнения таблицы  $\epsilon = K/N$ . Использовать  $N = 10\,007$  и  $N = 257$ . Сделайте вывод о полезности двойного хеширования, сравнив результаты со случаем, когда  $h_2(K) = 1$ .

1.17. «Неприкасаемый король». На поле С3 шахматной доски неподвижно стоит белый король. Затем на доске устанавливаются имеющие возможность двигаться белый ферзь – на любую свободную клетку и черный король – на любую возможную для него клетку. Доказать, что для любой «возможной и разумной\*» начальной позиции мат черному королю может быть объявлен за число ходов меньше 25 (независимо от того, чьим является первый ход).

\*Невозможной является, например, позиция, когда два короля оказались на соседних клетках, а неразумной, – когда на соседних клетках оказались черный король и незащищенный белый ферзь.)