УПРАВЛЕНИЕ ОБРАЗОВАНИЯ АДМИНИСТРАЦИИ ГОРОДА ВОЛОГДЫ
МУНИЦИПАЛЬНОЕ УЧРЕЖДЕНИЕ ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
«ДЕТСКО-ЮНОШЕСКИЙ ЦЕНТР «ЕДИНСТВО»

Рассмотрено на педагогическом совете
МУ ДО «ДЮЦ «Единство»
Протокол № 4 от 31 мая 2023 г.

УТВЕРЖДАЮ
Директор МУ ДО «ДЮЦ «Единство»
Н.В. Шадрина
Приказ № 86 от 31 мая 2023 года

# ДОПОЛНИТЕЛЬНАЯ ОБЩЕОБРАЗОВАТЕЛЬНАЯ ОБЩЕРАЗВИВАЮЩАЯ ПРОГРАММА

технической направленности

# «ФУНДАМЕНТАЛЬНЫЕ АЛГОРИТМЫ»

Уровень программы: углублённый

Возраст детей: 14 -17 лет
Срок реализации: 9 месяцев
Объем программы: 144 часа

Автор-составитель:
Назаров Николай Васильевич,
педагог дополнительного образования
МУ ДО ДЮЦ «Единство»

Вологда
2023

## ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Дополнительная общеобразовательная общеразвивающая программа «Фундаментальные алгоритмы» имеет *техническую направленность.*

Программа разработана в соответствии со следующими нормативными документами:

— Федеральный закон «Об образовании в Российской Федерации» от 29 декабря 2012 года № 273-ФЗ;

— Стратегия развития воспитания в РФ на период до 2025 года, утверждена Распоряжением Правительства РФ от 29 мая 2015 г. № 996-р;

— Государственная программа РФ «Развитие образования», утверждена Постановлением Правительства РФ от 26 декабря 2017 года № 1642;

— Федеральный проект «Успех каждого ребенка», утвержденный президиумом Совета при Президенте РФ по стратегическому развитию и национальным проектам (протокол от 24 декабря 2018 года № 16);

— Приказ Министерства просвещения РФ от 9 ноября 2018 г. N 196 «Об утверждении Порядка организации и осуществления образовательной деятельности по дополнительным общеобразовательным программам»;

— Приказ Министерства просвещения РФ от 3 сентября 2019 г. № 467 «Об утверждении Целевой модели развития региональных систем дополнительного образования детей»;

— Методические рекомендации по проектированию дополнительных общеразвивающих программ (включая разноуровневые программы). Письмо Министерства образования и науки России от 18 ноября 2015 года №09-3242;

— Распоряжение Правительства РФ от 31.03.2022 N 678-р «Об утверждении Концепции развития дополнительного образования детей до 2030 года».

**Актуальность программы**

В течение очень долгого временного промежутка содержание дисциплины «Информатика» не может устояться. За достаточно небольшое отведённое учебное время школьная информатика решает различные образовательные задачи, учебно-материальная база данной дисциплины меняется даже в течение года. Это привело к тому, что для проведения олимпиад по данному предмету был выделен достаточно узкий раздел информатики, связанный с алгоритмизацией и программированием. Фактически, начиная со школьных и заканчивая всероссийскими олимпиадами, все предлагаемые задания связаны с разработкой алгоритмов для решения задач, применением и реализацией этих алгоритмов на ЭВМ на одном из языков программирования. При этом редко происходит акцентирование на создание модели, с помощью которой решается та или иная

задача. Следует иметь в виду, что теория алгоритмов, теория их сложности и эффективности, методы их реализации на ЭВМ достаточно хорошо описаны в научной и учебной литературе, разбираются на различных интернет-ресурсах, но совершенно не отражены в курсе школьной информатики. То есть для того, чтобы школьник имел возможность результативно участвовать в олимпиадах по информатике, необходимы дополнительные задания, ориентированные на термин алгоритма.

Из-за такого обособленного положения олимпиад по информатике (а на деле, можно сказать, по программированию) курс математики оказывает на них ничуть не меньшее влияние, чем собственно курс информатики. Для большинства предлагаемых на подобных олимпиадах задач основную сложность несёт не реализация решения на ЭВМ, а предварительная разработка алгоритма решения «на бумаге» — создание или поиск подходящей комбинации из олимпиадных идей. Отсюда наблюдается крайняя зависимость успехов школьников от их знаний в области математики, особенно её олимпиадной поддисциплины. Можно констатировать, что олимпиадное программирование является родственным олимпиадной математике разделом с исключенной необходимостью доказывать справедливость решения, но с обязательной их реализацией на ЭВМ. Сведения о системах счисления, математической логике, вычислительной геометрии, теории чисел, комбинаторике, дискретной математике, теории графов, динамическом программировании рассматриваются в олимпиадном программировании в равной степени с олимпиадной математикой.

Вследствие рассмотренных выше обстоятельств (узость дисциплины, её уникальное положение на стыке математики и информатики, минимальное внимание алгоритмизации в школьном курсе), а также огромного числа идей, выходящих за пределы школьных курсов математики и информатики, малого числа специалистов достаточной квалификации (в основном преподаватель должен в прошлом или настоящем быть олимпиадником-программистом), высокого порога вхождения в дисциплину и сильной необходимости в самообучении олимпиадное программирование рекомендуется изучать именно в рамках центров дополнительного образования с последующим выходом на самообучение. В свою очередь в общеобразовательных учреждениях программирование как учебный предмет по-прежнему проходит этап становления, ещё ведутся дискуссии по поводу его содержания.

Важнейшим приоритетом дополнительного образования в условиях становления глобального информационного общества становится формирование у обучающихся представлений о возможностях программирования для развития логического мышления, способностей к формализации, элементов системного мышления и воспитания чувства ответственности за результаты своего труда,

установки на недопустимость действий, нарушающих правовые, этические нормы; развития творческих способностей и самостоятельности при решении различных вопросов.

В первую очередь следует рассматривать программирование как средство развития мышления школьников. Знание языка программирования, как и владение любым другим инструментом, само по себе ничего не даёт, куда важнее умение им пользоваться. Только при самостоятельном решении задач можно говорить о развитии у обучающихся способности принятия решения и ответственности за последствия. Начиная с определённого момента, успехи школьника определяются именно количеством идей и приёмов, которыми обучающийся успел овладеть. Для достижения максимального результата большую часть знаний обучающемуся предстоит освоить самостоятельно, индивидуально. Особое внимание в курсе уделяется обучению школьников помимо теоретического базиса и основным умениям и навыкам, которые потребуются для успешной и максимально эффективной самостоятельной работы.

Другим важным вопросом является понимание единства информационных принципов строения и функционирования самоуправляемых систем различной природы, процессов управления в природе, технике, социуме. Очень важным моментом является задача инициировать самостоятельную познавательную деятельность обучающихся, заставлять их постоянно задумываться над «природой вещей», сообщив лишь небольшое количество исходной информации.

«Фундаментальные алгоритмы» рассматривается как *специфический курс, непрерывно развивающий знания школьников в области программирования и информационно-коммуникационных технологий и тесной интеграции с математикой.* Обучение по программе обеспечит подготовку школьников к интеграции в высокотехнологичную среду.

Олимпиады по программированию – довольно молодое направление в школьном олимпиадном движении. Основой для достижения высоких результатов служит целенаправленная подготовка, изучение основных идей и принципов программирования, освоение наиболее известных приемов (алгоритмов), освоение их реализации на ЭВМ на языке программирования.

За последнее время накоплен достаточно большой опыт участия российских школьников и студентов в олимпиадах по информатике и программированию различного уровня. Анализ этого опыта позволил очертить примерный список тем и вопросов, которые следует изучить, чтобы подготовиться к участию в олимпиадах по программированию.

**Цели программы:**

— Развитие логического, алгоритмического и латерального мышления

обучающихся в процессе решения олимпиадных задач по программированию в интеграции с математикой;

— Достижение уровня, достаточного для эффективного самообучения в области олимпиадного программирования, и последующее дальнейшее развитие в этой области.

**Задачи программы:**

*1. Воспитательные (личностные)*

— формирование личностного смысла усвоения знаний (ценностно-смысловые установки, цифровая этика, мировоззрение, профориентация, мотивация);

— развитие активности обучающегося на основе мотивации к включению в активную познавательную деятельность;

— воспитание самостоятельности и дисциплинированности в информационных потоках, развитие мотивированности к самостоятельному познанию и творчеству;

— воспитание чувства ответственности за результаты своей деятельности;

— развитие умений, требуемых для эффективной работы в команде;

— развитие навыков сотрудничества со взрослыми и сверстниками в разных социальных ситуациях;

— формирование культуры информационного, виртуального общения в социуме;

— формирование навыков здорового образа жизни через понимание правил работы с компьютером и информационными потоками;

— профориентация и формирование готовности обучающихся к профессиональному самоопределению — подготовка к поступлению в высшие учебные заведения.

*2. Метапредметные*

— формирование основ самооценки и самоконтроля;

— развитие навыков, требуемых для эффективной организации самостоятельной работы;

— развитие логического мышления, способностей к формализации;

— формирование умения грамотно и структурированно излагать свои идеи и толкования;

— развитие навыков коммуникации со взрослыми и сверстниками в ходе решения олимпиадных задач для дальнейшего использования данного опыта в различных социальных обстановках;

— развитие творческих способностей в процессе алгоритмизации в рамках учебного курса, развитие навыков принятия как очевидных, единственно верных, так и нестандартных решений в учебной деятельности и повседневных

обстановках;

— подготовка обучающихся к восприятию большого потока новой информации;

— развитие деловых качеств: независимости с одной стороны и ответственности, аккуратности – с другой;

— развитие познавательных и интеллектуальных интересов.

*3. Предметные*

— систематизация имеющихся знаний и получение новых в областях математики, программирования, информационно-коммуникационных технологий;

— формирование умений по разработке программных продуктов;

— формирование культуры работы с проектами и культуры оформления программного кода;

— знакомство с особенностями методологии исследовательских и проектных работ в области программирования и информационно-коммуникационных технологий;

— опережение сверстников в таких областях знаний как математика, программирование и информационно-коммуникационные технологии;

— подготовка к выступлениям на соревнованиях разных уровней.

**Отличительные особенности программы**

Программа *вариативная,* так как в рамках ее содержания можно разрабатывать разные учебно-тематические планы и для ее освоения возможно выстраивание индивидуальных программ, индивидуальных траекторий (маршрутов) обучения. Программа *открытая,* предполагает совершенствование, изменение в соответствии с потребностями обучающихся и тенденциями развития этой области знаний.

Особенности содержания программы следующие: с одной стороны *фундаментализация, усиление системности и функциональности теоретических знаний*; с другой — *повышение внимания к прикладной стороне содержания, его методологической и практической направленности*.

При этом в программе учтена тесная взаимосвязь математики и программирования: отдельное внимание уделяется основополагающим олимпиадно-математическим идеям и их применению в решении реальных задач.

Используемые в программе задачи взяты из различных источников, ими могут являться как олимпиады по программированию разных уровней, так и составленные разработчиками самостоятельно задачи. Большая часть задач взята с сайта Codeforces, сайта дистанционной подготовки по информатике МЦНМО «Информатикс», сайта интернет-олимпиад по информатике и программированию г. Санкт-Петербург, сайта Timus Online Judge, сайта Школа Программиста

(который в свою очередь тоже использует задачи уже озвученных ресурсов, а также ресурс Олимпиады по программированию и книгу Федора Меньшикова «Олимпиадные задачи по программированию»).

Настоящая программа рассчитана на работу в детском объединении в системе дополнительного образования.

**В основу курса положены такие принципы как:**

— *Целостность и непрерывность*, означающие, что данная ступень является важным звеном единой подготовки школьников в сфере математики и информационных технологий

— *Принцип эвристического обучения* (ставящий целью конструирование обучающимся собственного смысла и содержания образования, а также процесса его организации, диагностики и осознания; кроме того ставящий при этом целью непрерывное открытие обучающимся чего-то нового)

— *Принцип развивающего обучения* (обучение ориентировано не только на получение новых знаний в области математики и программирования, но и на активизацию мыслительных процессов, формирование и развитие у обучающихся обобщенных способов деятельности, формирование навыков самостоятельной работы в цифровой среде)

— *Практико-ориентированность*, обеспечивающая отбор содержания, направленного на решение практических задач. При этом исходным является положение о том, что компьютер может многократно усилить возможности человека, но не заменить его

— *Научность в сочетании с доступностью, строгость и систематичность изложения* (включение в содержание фундаментальных положений современной науки с учетом возрастных особенностей обучаемых)

**Уровень программы:**

Углублённый (продвинутый)

**Адресат программы:**

Программа предназначена для обучающихся возрастом 14–17 лет, уже проходивших обучение по дополнительным общеобразовательным программам в области олимпиадного или алгоритмического программирования. Является наиболее комплексной (сложной) программой объединения, адресованной в первую очередь «олимпиадникам», готовящимся к соревнованиям по программированию высокого уровня. Допускается совместное изучение материала школьниками смежных возрастов

**Срок освоения программы:**

9 месяцев, 36 недель — *программа реализуется в течение всего*

*календарного года, включая каникулярное время, с 1 сентября по 31 мая*

**Формы и виды занятий:**

Обучение очное с элементами дистанционного обучения. Для успешного освоения обучающимся программы предполагается домашнее дорешивание задач, разбираемых в рамках объединения

Формы занятий: лекции, индивидуальные консультации, групповые консультации, тематические дискуссии, круглые столы, индивидуальные практические занятия, олимпиады *(в том числе групповые, т. е. командные)*

**Режим занятий:**

Занятия проводятся в очной форме еженедельно. Занятия проводятся два раза в неделю по два академических часа. Продолжительность одного академического часа 40 минут, с перерывом 10 минут

**Численность обучающихся в группе:**

10 – 12 человек

**Объём программы:**

Распределение часов по темам дано из расчета максимум 144 часа в год — 57 часа теории и 87 час практических занятий

# СОДЕРЖАНИЕ ПРОГРАММЫ

## Учебный план

| № п/п | Тематический блок | Часов теор. | Часов практ | Итого часов | Форма контроля |
|---|---|---|---|---|---|
| I | Геометрия в программировании | 3 | 5 | 8 | Комбинированная: *письменный опрос и программный продукт* |
| II | Длинная арифметика | 1 | 3 | 4 | Программный продукт |
| III | Расширение отдельных изученных олимпиадных идей | 6 | 6 | 12 | Комбинированная: *письменный опрос и программный продукт* |
| IV | Динамическое программирование | 6 | 14 | 20 | Программный продукт |
| V | Алгоритмы на графах. Кратчайшие пути | 3 | 7 | 10 | Программный продукт |
| VI | Алгоритмы на графах. Поиск в глубину и его применения | 2 | 8 | 10 | Программный продукт |
| VII | Алгоритмы на строках | 6 | 10 | 16 | Программный продукт |
| VIII | Алгебра, теория чисел и дискретная математика | 10 | 10 | 20 | Письменный опрос |
| IX | Структуры данных | 14 | 16 | 30 | Комбинированная: *письменный опрос и программный продукт* |
| X | Графы-деревья | 6 | 8 | 14 | Комбинированная: *письменный опрос и программный продукт* |
| | **ИТОГО:** | **57** | **87** | **144** | |

**Учебно-тематический план**

| № п/п | Тема / Тематический блок | Часов теор. | Часов практ | Итого часов |
|---|---|---|---|---|
| **I** | **Геометрия в программировании** | **3** | **5** | **8** |
| 1 | Основные объекты: точка, отрезок, луч, угол, многоугольник, окружность и прочие. Геометрия в координатах. Идея симметрии (центральная, осевая), параллельного переноса, вращения. Взаимные расположения точек и прямых | | 2 | |
| 2 | Геометрия векторов. Скалярное произведение. Косое произведение. Использование косого произведения для нахождения ориентированной площади | 1 | 1 | |
| 3 | Проблемы, возникающие при работе с числами с плавающей точкой, сравнение подобных чисел. Взаимное расположении точки и многоугольника: метод площадей, метод трассировки лучей, метод ближайшей точки. Формула Пика нахождения площадки решётчатого многоугольника | 1 | 1 | |
| 4 | Выпуклые многоугольники. Построение выпуклой оболочки для набора точек | 1 | 1 | |
| **II** | **Длинная арифметика** | **1** | **3** | **4** |
| 5 | Понятие длинного числа. Хранение и изменение длинных чисел. Умножение длинного числа на короткое. Сложение, сравнение и разность длинных чисел | 1 | 1 | |
| 6 | Произведение длинных чисел. Деление длинного числа на короткое с остатком. Реализация своего интерфейса длинного числа | | 2 | |
| **III** | **Расширение отдельных изученных олимпиадных идей** | **6** | **6** | **12** |
| 7 | Конструктив и задачи на эффективную реализацию. Жадные алгоритмы и сортировки | | 2 | |
| 8 | Линейные алгоритмы. Использование частичных сумм. Задача поиска подотрезка массива с максимальной суммой. Задача нахождения K-й порядковой статистики | 1 | 1 | |
| 9 | Введение в дискретную математику. Основы булевой алгебры. Побитовые операции в программировании. Возможности их применения | 2 | | |

| № п/п | Тема / Тематический блок | Часов теор. | Часов практ | Итого часов |
|---|---|---|---|---|
| 10 | Перебор. Применение побитовых операций для полного перебора подмножеств. Битовые маски | 1 | 1 | |
| 11 | Два указателя и двоичный поиск. Двоичный поиск по ответу. Разбор более трудных задач, решаемых двоичным поиском по ответу | 1 | 1 | |
| 12 | Алгоритм тернарного поиска экстремума унимодальной функции | 1 | 1 | |
| **IV** | **Динамическое программирование** | **6** | **14** | **20** |
| 13 | Теоретическая основа динамического программирования. Оптимальная структура. Принцип оптимальности на префиксе | 1 | 1 | |
| 14 | Простейшие задачи двумерного ДП. Задачи на подсчёт количества путей в табличках и лабиринтах | | 2 | |
| 15 | Простейшие задачи двумерного ДП. Задачи на подсчёт количества или существования комбинаций с определёнными ограничениями на их построение. Выбор аргументов для динамического программирования | | 2 | |
| 16 | Принцип оптимальности на подотрезках. Задача расстановки знаков в выражении. Задача нахождения расстояния Левенштейна | 1 | 1 | |
| 17 | ДП на подотрезках. Задача наибольшей общей подпоследовательности. Задача наибольшей подпоследовательности-палиндрома | | 2 | |
| 18 | Использование двоичного поиска в динамическом программировании для улучшения асимптотики. Оптимизация задачи наибольшей возрастающей подпоследовательности. Задача наибольшей общей возрастающей подпоследовательности | 1 | 1 | |
| 19 | Использование топологической сортировки. Динамическое программирование на графах. Кратчайший путь в ациклическом графе. Количество путей в ациклическом графе | 1 | 1 | |
| 20 | Простейшие задачи ДП на подмножествах | | 2 | |
| 21 | Задача о 0-1 рюкзаке. Задача о неограниченном рюкзаке | 1 | 1 | |

| № п/п | Тема / Тематический блок | Часов теор. | Часов практ | Итого часов |
|---|---|---|---|---|
| 22 | Динамическое программирование по профилю. Задачи замощения доски доминошками | 1 | 1 | |
| **V** | **Алгоритмы на графах. Кратчайшие пути** | **3** | **7** | **10** |
| 23 | Алгоритм Флойда-Уоршелла нахождения всевозможных кратчайших путей. Его использование на невзвешенном графе. Идея веса ребра, алгоритм Флойда-Уоршелла на взвешенном графе | 1 | 1 | |
| 24 | Обход в ширину на 0-1 графе. Обход в ширину на 1..k-графе с натуральным весом рёбер | | 2 | |
| 25 | Алгоритм Дейкстры нахождения кратчайшего взвешенного пути. Алгоритм Дейкстры с логарифмической асимптотической сложностью | 1 | 1 | |
| 26 | Хранение графа списком рёбер. Алгоритм Беллмана-Форда нахождения кратчайшего взвешенного пути. Нахождение цикла отрицательного веса | 1 | 1 | |
| 27 | Решение различных задач на кратчайшие пути на взвешенных графах | | 2 | |
| **VI** | **Алгоритмы на графах. Поиск в глубину и его применения** | **2** | **8** | **10** |
| 28 | Отношение сильной связности. Разбиение ориентированного графа на компоненты сильной связности | 1 | 1 | |
| 29 | Задачи на применение разбиения на компоненты сильной связности. Построение конденсации графа и её применение | | 2 | |
| 30 | Вершинная и рёберная связность. Важные вершины и рёбра: точки сочленения и мосты. Изучение свойств дерева обхода в глубину, лемма о белых путях | 1 | 1 | |
| 31 | Применение алгоритма поиска мостов | | 2 | |
| 32 | Применение алгоритма поиска точек сочленения | | 2 | |
| **VII** | **Алгоритмы на строках** | **6** | **10** | **16** |
| 33 | Строки из ЯП Си. Указатели. Работа с си-строками, функции библиотеки <cstring> | 1 | 1 | |

| № п/п | Тема / Тематический блок | Часов теор. | Часов практ | Итого часов |
|---|---|---|---|---|
| 34 | Разбор строк и лексический анализ. Перечисления в C++, шаблон «enum» | 1 | 1 | |
| 35 | Синтаксический анализ. Метод рекурсивного спуска | 1 | 1 | |
| 36 | Задача поиска подстроки в строке. Наивный алгоритм и возможности его улучшения. Префиксы и суффиксы | | 2 | |
| 37 | Z-функция и её эффективное вычисление | 1 | 1 | |
| 38 | Алгоритм Кнута-Мориса-Пратта | | 2 | |
| 39 | Префикс-функция и её эффективное вычисление. Построение Z-функции по префикс-функции и наоборот | 1 | 1 | |
| 40 | Решение задачи количества уникальных подстрок в строке с помощью префикс-функции | 1 | 1 | |
| **VIII** | **Алгебра, теория чисел и дискретная математика** | **10** | **10** | **20** |
| 41 | Матрицы в линейной алгебре. Сложение, умножение матриц, возведение квадратной матрицы в натуральную степень. Определитель квадратной матрицы | 2 | | |
| 42 | Бинарное возведение числа в натуральную степень. Бинарное возведение квадратной матрицы в натуральную степень. Бинарный алгоритм Евклида | | 2 | |
| 43 | Применение матриц в программировании. Количество путей фиксированной длины в графе. Быстрое вычисление K-го числа Фибоначчи | | 2 | |
| 44 | Расширенный алгоритм Евклида. Линейные диофантовы уравнения с двумя переменными | 1 | 1 | |
| 45 | Кольцо вычетов по модулю. Нахождение обратного элемента в кольце вычетов по модулю | 1 | 1 | |
| 46 | Китайская теорема об остатках. Алгоритм Гарнера. Функция Эйлера и её вычисление | 2 | | |
| 47 | Коды Грея. Коды Грея для перестановок. Задача о Ханойских башнях | 1 | 1 | |
| 48 | Полиномиальный хеш, хеш-функция. Предподсчёт в алгоритме хеширования. Вычисление хеша подстроки | 1 | 1 | |

| № п/п | Тема / Тематический блок | Часов теор. | Часов практ | Итого часов |
|---|---|---|---|---|
| 49 | Коллизии и их минимизация. Хеш-таблица как способ реализации ассоциативного массива. Методы разрешения коллизий в хеш-таблице | 2 | | |
| 50 | Алгоритм Рабина-Карпа поиска подстроки в строке. Решение задачи количества уникальных подстрок в строке с помощью хеширования. Функция «std::unique» | | 2 | |
| **IX** | **Структуры данных** | **14** | **16** | **30** |
| 51 | Двоичное дерево поиска. Сбалансированные деревья, двоичные кучи. Биномиальная куча, фибоначчиева куча. Приоритетные очереди | 2 | | |
| 52 | Реализация двоичной кучи и её использование в задачах | | 2 | |
| 53 | Декартово дерево. Декартово дерево по неявному ключу | 1 | 1 | |
| 54 | Идея "+1/–1" массовой модификации подотрезка за O(1) | 1 | 1 | |
| 55 | Очередь с хранением локального максимума / минимума для его нахождения за O(1) | 1 | 1 | |
| 56 | Задачи RMQ, RSQ. Корневая эвристика | 1 | 1 | |
| 57 | Корневая эвристика с операциями удаления и вставки в массив | 1 | 1 | |
| 58 | Алгоритм Мо | 1 | 1 | |
| 59 | Система непересекающихся множеств. Наивная реализация, эвристики для её улучшения | 2 | | |
| 60 | Остовное дерево. Минимальное остовное дерево, алгоритмы его нахождения. Алгоритм Краскала. Алгоритм Прима | 1 | 1 | |
| 61 | Решение задач на остовные деревья и СНМ | | 2 | |
| 62 | Решение задачи RMQ с помощью разреженной таблицы | 1 | 1 | |
| 63 | Дерево отрезков: построение и реализация запросов | 1 | 1 | |
| 64 | Решение задач RMQ, RSQ, RGQ и других с помощью построения деревьев отрезков | | 2 | |
| 65 | Операция массового обновления на дереве отрезков. Ленивое распространение | 1 | 1 | |

| № п/п | Тема / Тематический блок | Часов теор. | Часов практ | Итого часов |
|---|---|---|---|---|
| **X** | **Графы-деревья** | **6** | **8** | **14** |
| 66 | Особенности алгоритмов обхода для деревьев. "Подвешивание" дерева, корневые деревья. Распространение на корневом дереве | 1 | 1 | |
| 67 | Нахождение диаметра дерева. Центральные вершины дерева. Их использование в решении задач | | 2 | |
| 68 | Модификации алгоритма обхода в глубину на дереве. Подсчёт размеров поддеревьев, порядка входа и выхода из вершин. Задача о сумме чисел в поддереве с модификацией. Проверка на предка за O(1) | 1 | 1 | |
| 69 | Ближайший общий предок. Эйлеров обход дерева и сведение задачи LCA к задаче RMQ и дереву отрезков | 1 | 1 | |
| 70 | Метод двоичного подъёма с динамическим препроцессингом. Нахождение ближайшего общего предка методом двоичного подъёма | 1 | 1 | |
| 71 | Нахождение центроида дерева. Центроидная декомпозиция | 1 | 1 | |
| 72 | Динамическое программирование на поддеревьях. Использование LCA в ДП на поддеревьях. Расширение идеи "+1/–1" для массовой модификации простых цепей на дереве | 1 | 1 | |

**Содержание учебно-тематического плана**

**I. Геометрия в программировании**

Теория: Геометрия векторов. Скалярное произведение. Косое произведение. Использование косого произведения для нахождения ориентированной площади. Проблемы, возникающие при работе с числами с плавающей точкой, сравнение подобных чисел. Взаимное расположении точки и многоугольника: метод площадей, метод трассировки лучей, метод ближайшей точки. Формула Пика нахождения площадки решётчатого многоугольника. Выпуклые многоугольники. Построение выпуклой оболочки для набора точек

Практические занятия:

1. Основные объекты: точка, отрезок, луч, угол, многоугольник, окружность и

прочие. Геометрия в координатах. Идея симметрии (центральная, осевая), параллельного переноса, вращения. Взаимные расположения точек и прямых
2. Геометрия векторов. Скалярное произведение. Косое произведение. Использование косого произведения для нахождения ориентированной площади
3. Проблемы, возникающие при работе с числами с плавающей точкой, сравнение подобных чисел. Взаимное расположении точки и многоугольника: метод площадей, метод трассировки лучей, метод ближайшей точки. Формула Пика нахождения площадки решётчатого многоугольника
4. Выпуклые многоугольники. Построение выпуклой оболочки для набора точек

**II. Длинная арифметика**

Теория: Понятие длинного числа. Хранение и изменение длинных чисел. Умножение длинного числа на короткое. Сложение, сравнение и разность длинных чисел

Практические занятия:
1. Понятие длинного числа. Хранение и изменение длинных чисел. Умножение длинного числа на короткое. Сложение, сравнение и разность длинных чисел
2. Произведение длинных чисел. Деление длинного числа на короткое с остатком. Реализация своего интерфейса длинного числа

**III. Расширение отдельных изученных олимпиадных идей**

Теория: Линейные алгоритмы. Использование частичных сумм. Задача поиска подотрезка массива с максимальной суммой. Задача нахождения K-й порядковой статистики. Введение в дискретную математику. Основы булевой алгебры. Побитовые операции в программировании. Возможности их применения. Перебор. Применение побитовых операций для полного перебора подмножеств. Битовые маски. Два указателя и двоичный поиск. Двоичный поиск по ответу. Разбор более трудных задач, решаемых двоичным поиском по ответу. Алгоритм тернарного поиска экстремума унимодальной функции

Практические занятия:
1. Конструктив и задачи на эффективную реализацию. Жадные алгоритмы и сортировки
2. Линейные алгоритмы. Использование частичных сумм. Задача поиска подотрезка массива с максимальной суммой. Задача нахождения K-й порядковой статистики
3. Перебор. Применение побитовых операций для полного перебора

подмножеств. Битовые маски

4. Два указателя и двоичный поиск. Двоичный поиск по ответу. Разбор более трудных задач, решаемых двоичным поиском по ответу
5. Алгоритм тернарного поиска экстремума унимодальной функции

## IV. Динамическое программирование

Теория: Теоретическая основа динамического программирования. Оптимальная структура. Принцип оптимальности на префиксе. Принцип оптимальности на подотрезках. Задача расстановки знаков в выражении. Задача нахождения расстояния Левенштейна. Использование двоичного поиска в динамическом программировании для улучшения асимптотики. Оптимизация задачи наибольшей возрастающей подпоследовательности. Задача наибольшей общей возрастающей подпоследовательности. Использование топологической сортировки. Динамическое программирование на графах. Кратчайший путь в ациклическом графе. Количество путей в ациклическом графе. Задача о 0-1 рюкзаке. Задача о неограниченном рюкзаке. Динамическое программирование по профилю. Задачи замощения доски доминошками

Практические занятия:

1. Теоретическая основа динамического программирования. Оптимальная структура. Принцип оптимальности на префиксе
2. Простейшие задачи двумерного ДП. Задачи на подсчёт количества путей в табличках и лабиринтах
3. Простейшие задачи двумерного ДП. Задачи на подсчёт количества или существования комбинаций с определёнными ограничениями на их построение. Выбор аргументов для динамического программирования
4. Принцип оптимальности на подотрезках. Задача расстановки знаков в выражении. Задача нахождения расстояния Левенштейна
5. ДП на подотрезках. Задача наибольшей общей подпоследовательности. Задача наибольшей подпоследовательности-палиндрома
6. Использование двоичного поиска в динамическом программировании для улучшения асимптотики. Оптимизация задачи наибольшей возрастающей подпоследовательности. Задача наибольшей общей возрастающей подпоследовательности
7. Использование топологической сортировки. Динамическое программирование на графах. Кратчайший путь в ациклическом графе. Количество путей в ациклическом графе
8. Простейшие задачи ДП на подмножествах
9. Задача о 0-1 рюкзаке. Задача о неограниченном рюкзаке

10. Динамическое программирование по профилю. Задачи замощения доски доминошками

### V. Алгоритмы на графах. Кратчайшие пути

Теория: Алгоритм Флойда-Уоршелла нахождения всевозможных кратчайших путей. Его использование на невзвешенном графе. Идея веса ребра, алгоритм Флойда-Уоршелла на взвешенном графе. Алгоритм Дейкстры нахождения кратчайшего взвешенного пути. Алгоритм Дейкстры с логарифмической асимптотической сложностью. Хранение графа списком рёбер. Алгоритм Беллмана-Форда нахождения кратчайшего взвешенного пути. Нахождение цикла отрицательного веса

Практические занятия:

1. Алгоритм Флойда-Уоршелла нахождения всевозможных кратчайших путей. Его использование на невзвешенном графе. Идея веса ребра, алгоритм Флойда-Уоршелла на взвешенном графе
2. Обход в ширину на 0-1 графе. Обход в ширину на 1..k-графе с натуральным весом рёбер
3. Алгоритм Дейкстры нахождения кратчайшего взвешенного пути. Алгоритм Дейкстры с логарифмической асимптотической сложностью
4. Хранение графа списком рёбер. Алгоритм Беллмана-Форда нахождения кратчайшего взвешенного пути. Нахождение цикла отрицательного веса
5. Решение различных задач на кратчайшие пути на взвешенных графах

### VI. Алгоритмы на графах. Поиск в глубину и его применения

Теория: Отношение сильной связности. Разбиение ориентированного графа на компоненты сильной связности. Вершинная и рёберная связность. Важные вершины и рёбра: точки сочленения и мосты. Изучение свойств дерева обхода в глубину, лемма о белых путях

Практические занятия:

1. Отношение сильной связности. Разбиение ориентированного графа на компоненты сильной связности
2. Задачи на применение разбиения на компоненты сильной связности. Построение конденсации графа и её применение
3. Вершинная и рёберная связность. Важные вершины и рёбра: точки сочленения и мосты. Изучение свойств дерева обхода в глубину, лемма о белых путях
4. Применение алгоритма поиска мостов

5. Применение алгоритма поиска точек сочленения

### VII. Алгоритмы на строках

Теория: Строки из ЯП Си. Указатели. Работа с си-строками, функции библиотеки <cstring>. Разбор строк и лексический анализ. Перечисления в C++, шаблон «enum». Синтаксический анализ. Метод рекурсивного спуска. Z-функция и её эффективное вычисление. Префикс-функция и её эффективное вычисление. Построение Z-функции по префикс-функции и наоборот. Решение задачи количества уникальных подстрок в строке с помощью префикс-функции

Практические занятия:

1. Строки из ЯП Си. Указатели. Работа с си-строками, функции библиотеки <cstring>
2. Разбор строк и лексический анализ. Перечисления в C++, шаблон «enum»
3. Синтаксический анализ. Метод рекурсивного спуска
4. Задача поиска подстроки в строке. Наивный алгоритм и возможности его улучшения. Префиксы и суффиксы
5. Z-функция и её эффективное вычисление
6. Алгоритм Кнута-Мориса-Пратта
7. Префикс-функция и её эффективное вычисление. Построение Z-функции по префикс-функции и наоборот
8. Решение задачи количества уникальных подстрок в строке с помощью префикс-функции

### VIII. Алгебра, теория чисел и дискретная математика

Теория: Матрицы в линейной алгебре. Сложение, умножение матриц, возведение квадратной матрицы в натуральную степень. Определитель квадратной матрицы. Расширенный алгоритм Евклида. Линейные диофантовы уравнения с двумя переменными. Кольцо вычетов по модулю. Нахождение обратного элемента в кольце вычетов по модулю. Китайская теорема об остатках. Алгоритм Гарнера. Функция Эйлера и её вычисление. Коды Грея. Коды Грея для перестановок. Задача о Ханойских башнях. Полиномиальный хеш, хеш-функция. Предподсчёт в алгоритме хеширования. Вычисление хеша подстроки. Коллизии и их минимизация. Хеш-таблица как способ реализации ассоциативного массива. Методы разрешения коллизий в хеш-таблице

Практические занятия:

1. Бинарное возведение числа в натуральную степень. Бинарное возведение квадратной матрицы в натуральную степень. Бинарный алгоритм Евклида

2. Применение матриц в программировании. Количество путей фиксированной длины в графе. Быстрое вычисление K-го числа Фибоначчи
3. Расширенный алгоритм Евклида. Линейные диофантовы уравнения с двумя переменными
4. Кольцо вычетов по модулю. Нахождение обратного элемента в кольце вычетов по модулю
5. Коды Грея. Коды Грея для перестановок. Задача о Ханойских башнях
6. Полиномиальный хеш, хеш-функция. Предподсчёт в алгоритме хеширования. Вычисление хеша подстроки
7. Алгоритм Рабина-Карпа поиска подстроки в строке. Решение задачи количества уникальных подстрок в строке с помощью хеширования. Функция «std::unique»

## IX. Структуры данных

Теория: Двоичное дерево поиска. Сбалансированные деревья, двоичные кучи. Биномиальная куча, фибоначчиева куча. Приоритетные очереди. Декартово дерево. Декартово дерево по неявному ключу. Идея "+1/–1" массовой модификации подотрезка за O(1). Очередь с хранением локального максимума / минимума для его нахождения за O(1). Задачи RMQ, RSQ. Корневая эвристика. Корневая эвристика с операциями удаления и вставки в массив. Алгоритм Мо. Система непересекающихся множеств. Наивная реализация, эвристики для её улучшения. Остовное дерево. Минимальное остовное дерево, алгоритмы его нахождения. Алгоритм Краскала. Алгоритм Прима. Решение задачи RMQ с помощью разреженной таблицы. Дерево отрезков: построение и реализация запросов. Операция массового обновления на дереве отрезков. Ленивое распространение

Практические занятия:
1. Реализация двоичной кучи и её использование в задачах
2. Декартово дерево. Декартово дерево по неявному ключу
3. Идея "+1/–1" массовой модификации подотрезка за O(1)
4. Очередь с хранением локального максимума / минимума для его нахождения за O(1)
5. Задачи RMQ, RSQ. Корневая эвристика
6. Корневая эвристика с операциями удаления и вставки в массив
7. Алгоритм Мо
8. Остовное дерево. Минимальное остовное дерево, алгоритмы его нахождения. Алгоритм Краскала. Алгоритм Прима
9. Решение задач на остовные деревья и СНМ

10. Решение задачи RMQ с помощью разреженной таблицы
11. Дерево отрезков: построение и реализация запросов
12. Решение задач RMQ, RSQ, RGQ и других с помощью построения деревьев отрезков
13. Операция массового обновления на дереве отрезков. Ленивое распространение

**X. Графы-деревья**

Теория: Особенности алгоритмов обхода для деревьев. "Подвешивание" дерева, корневые деревья. Распространение на корневом дереве. Модификации алгоритма обхода в глубину на дереве. Подсчёт размеров поддеревьев, порядка входа и выхода из вершин. Задача о сумме чисел в поддереве с модификацией. Проверка на предка за O(1). Ближайший общий предок. Эйлеров обход дерева и сведение задачи LCA к задаче RMQ и дереву отрезков. Метод двоичного подъёма с динамическим препроцессингом. Нахождение ближайшего общего предка методом двоичного подъёма. Нахождение центроида дерева. Центроидная декомпозиция. Динамическое программирование на поддеревьях. Использование LCA в ДП на поддеревьях. Расширение идеи "+1/–1" для массовой модификации простых цепей на дереве

Практические занятия:

1. Особенности алгоритмов обхода для деревьев. "Подвешивание" дерева, корневые деревья. Распространение на корневом дереве
2. Нахождение диаметра дерева. Центральные вершины дерева. Их использование в решении задач
3. Модификации алгоритма обхода в глубину на дереве. Подсчёт размеров поддеревьев, порядка входа и выхода из вершин. Задача о сумме чисел в поддереве с модификацией. Проверка на предка за O(1)
4. Ближайший общий предок. Эйлеров обход дерева и сведение задачи LCA к задаче RMQ и дереву отрезков
5. Метод двоичного подъёма с динамическим препроцессингом. Нахождение ближайшего общего предка методом двоичного подъёма
6. Нахождение центроида дерева. Центроидная декомпозиция
7. Динамическое программирование на поддеревьях. Использование LCA в ДП на поддеревьях. Расширение идеи "+1/–1" для массовой модификации простых цепей на дереве

## ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ

*1. Воспитательные (личностные)*

— осознанная познавательная деятельность как показатель сформированности ценностно-смысловых установок, отражающих личностные и гражданские позиции в деятельности;

— проявление активности обучающимся, мотивированное включение в активную познавательную деятельность;

— самостоятельность в планировании и дисциплинированность в осуществлении учебной деятельности;

— сформированное чувство ответственности за результаты своей деятельности;

— эффективная работа в командах на соревнованиях разных уровней, включая тренировочные (на занятиях);

— сформированные навыки сотрудничества со взрослыми и сверстниками в разных социальных ситуациях;

— усовершенствованная культура информационного и виртуального общения в социуме;

— знания в области здорового образа жизни через правила работы с компьютером и информационными потоками, понимание приоритетности ведения здорового образа жизни;

— сформированная профориентация или готовность к профессиональному самоопределению, ориентация на профессии актуальные для цифровой экономики.

*2. Метапредметные*

— владение навыками оценки и самоконтроля для продуктивной работы и взаимодействия;

— эффективная организация самостоятельной работы;

— заметные успехи в развитии логического мышления и способностей к формализации;

— структурированное изложение идей и толкований;

— усовершенствованные навыки коммуникации;

— способность к рационально-творческому мышлению: способность в зависимости от ситуации принимать как очевидные, единственно верные, так и нестандартные решения;

— готовность к восприятию больших потоков новой информации;

— развитые деловые качества: независимость с одной стороны и ответственность, аккуратность – с другой.

*3. Предметные*

— владение терминологией программирования в рамках программы;

— владение языком программирования C++ в рамках программы;

— владение олимпиадными идеями в рамках программы;

— эффективная работа с программным кодом;

— сформированные навыки алгоритмизации.

— наличие навыков, достаточных для разработки программных продуктов;

— наличие культуры работы с проектами и культуры оформления программного кода;

— знание основных правил и организацию олимпиад и контестов по программированию для школьников как заочной, так и очной формы;

— владение навыками учебно-исследовательской, проектной и социальной деятельности;

— опережение сверстников в таких областях знаний как математика, программирование и информационно-коммуникационные технологии;

— участие в соревнованиях разных уровней, достижение на них успехов.

**Основными критериями оценки эффективности образовательного процесса являются:**

— степень сформированности основных знаний, умений и навыков, предусмотренных программой;

— способность практически применять знания при создании программного продукта, при решении олимпиадных задач;

— личностный рост обучающихся;

— наличие у обучающихся личного смысла образования и собственного процесса его организации, их готовность к непрерывному открытию чего-то нового;

— успехи на соревнованиях разного уровня.

## ВОСПИТАТЕЛЬНЫЙ КОМПОНЕНТ

Особенностью программы является и компонентность образовательно-воспитательного процесса, взаимосвязь между ними:

I компонент — система дополнительного образования. Реализация дополнительной общеобразовательной общеразвивающей программы «Фундаментальные алгоритмы».

Целью первого компонента является формирование образовательного пространства и реализация в рамках образовательной программы дополнительного образования детей задач воспитания. При реализации программы взрослые выступают в роли педагогов дополнительного образования, наставников, педагогов – психологов, мастеров, а дети и подростки - в роли обучающихся, наставников (в системе «ребенок – ребенок»). В зависимости от темы, формы организации занятий строится адекватная система отношений, определяются нормы поведения в образовательном пространстве: ученичество, сотворчество и т.п.

II компонент - система воспитательных мероприятий. Предназначение второго компонента - обеспечение создания воспитательного пространства, в котором реализуются проекты, мероприятия и акции по основным направлениям воспитательной деятельности с использованием разнообразных форм организации.

*Календарный план воспитательной работы*

| **Название мероприятия, события** | **Форма проведения** | **Сроки** |
|---|---|---|
| День знаний | Беседа о роли знаний | 1 сентября |
| День программиста | Круглый стол «История и влияние профессии программиста на жизнь простого человека в наши дни» | 12/13 сентября |
| Неделя технического творчества | Серия интерактивных контестов «Компьютерная грамотность» | ноябрь-декабрь |
| Городская научно-практическая конференция «Мир науки» | Конференция (в соответствии с Положением) | январь-февраль |
| Всемирный день Земли | Беседа об угрозах окружающей среде, вызываемых массовым распространения электронной техники, и о возможностях позитивного влияния абсолютно каждого неравнодушного человека | 22 апреля |

## КОМПЛЕКС ОРГАНИЗАЦИОННО-ПЕДАГОГИЧЕСКИХ УСЛОВИЙ

**Условия реализации программы**

*1. Материально-техническое обеспечение:*

Для успешной реализации программы имеются: помещения, удовлетворяющие требованиям к образовательному процессу в учреждениях дополнительного образования, компьютеры (9-10), принтер и копировальный аппарат, мультимедиа, интернет.

*2. Кадровое обеспечение:*

Дополнительную образовательную программу реализуют педагоги дополнительного образования с классическим образованием в области программирования.

*3. Информационное обеспечение:*

— Сайт «Школа программиста» – https://acmp.ru

— Codeforces (*Онлайн-участие в соревнованиях, олимпиадах, контестах, отборах на очные туры*) – https://codeforces.com

— Информатикс, сайта дистанционной подготовки по информатике МЦНМО – https://informatics.msk.ru

— Topcoder (*Онлайн-участие в соревнованиях, олимпиадах, контестах, отборах на очные туры. На английском языке*) – https://www.topcoder.com

— Олимпиады школьников по информатике и программированию – https://neerc.ifmo.ru/school/information/index.html

— Олимпиады по программированию – https://olympiads.ru

— Timus Online Judge, архив задач с проверяющей системой – https://acm.timus.ru

— Вологодские областные и межрегиональные олимпиады по программированию – https://olympiads.vogu35.ru/interuni/index.php

— SPOJ, архив задач – https://www.spoj.com

— Викиконспекты — https://nerc.itmo.ru/mediawiki

— MAXimal, статьи по алгоритмам — https://e-maxx.ru/algo

**Формы аттестации и контроля**

Два раза в год в ходе *промежуточной аттестации* и *итогового контроля* осуществляется мониторинг результатов обучения и личностного развития в ходе освоения дополнительной образовательной программы.

Промежуточная аттестация и итоговый контроль проходят в форме олимпиады. Обучающимся предлагается небольшое число задач, предназначенных для решения на языке программирования на ЭВМ, также им предоставлена автоматизированная проверяющая система. По результатам

контроля обучающийся может получить одну из трёх оценок: незачёт, зачёт и зачёт с повышенным освоением программы (дифференцированный зачёт).

Для каждой задачи для проверки решений обучающихся заранее разработан полный пакет тестов, на которых будут запускаться написанные ими программы. Каждый тест включает в себя входные данные и правильный ответ на них. Один тест считается пройденным, если время, затраченное на выполнение, оказалось меньше установленного лимита, затраченный объём памяти также оказался меньше установленного лимита, а ответ программы учащегося для этих входных данных совпал с правильным. Если правильных ответов на тест может быть несколько, то имеется программа, которая умеет проверять конкретный ответ учащегося на правильность. Вся задача считается решённой тогда и только тогда, когда пройдены все тесты по данной задаче.

В качестве поощрения для наиболее успешно занимающихся школьников используются награждения по результатам их олимпиадной деятельности в течение года, поездки в лагеря, летние школы, на сборы, соревнования за пределы города и области, обеспечение необходимой литературой.

*Участие в олимпиадах* по информатике и программированию разного уровня является проверкой и степени освоения языка, и знания олимпиадных идей, и умения классифицировать предложенные на олимпиадах задачи для последующего применения полученных знаний на практике. Олимпиады (соревнования) по программированию зачастую являются командными, что также отлично позволяет оценить коммуникативные и организаторские способности участников.

*Участие в научно-практических конференциях и конкурсах* — на муниципальном, региональном, федеральном и международном уровнях позволяют оценить эффективность и степень освоения материала по проектной деятельности. Представление проектных работ допускается в форме устного доклада или презентации. Эта форма отчётности способствует формированию у обучающихся ответственности за выполнение работы, логики мышления, умения говорить перед аудиторией, отстаивать своё мнение, правильно использовать необходимую научную терминологию, корректно и грамотно вести дискуссию.

# ОЦЕНОЧНЫЕ МАТЕРИАЛЫ

**Системный мониторинг результативности обучения** по дополнительной общеразвивающей программе «Фундаментальные алгоритмы»

| Ожидаемый результат | Параметры | Критерии | Методы отслеживания |
|---|---|---|---|
| Умение принимать неочевидные решения, видеть нестандартные ходы как в учебной деятельности, так и в повседневной жизни | Изобретение способов решение проблем по красоте превосходящих авторские (общепринятые) | Статистика и красота, оригинальность таких решений | Анализ разрозненной информации |
| | | Количество человек, отмечающих изменения, произошедшие в ребенке | Педагогический консилиум |
| Значительное опережение сверстников в областях знаний, связанных с информатикой и программированием | Наличие обращений за помощью по предмету со стороны старших школьников и студентов | Количество обращений | Наблюдения учителей, беседа |
| | Успешность выступлений на соревнованиях | Количество побед на математических соревнованиях за более старшие классы (возрастные группы) | Анализ результатов соревнований |
| Умение эффективно работать над поставленной проблемой в коллективе | Соотношение коллективного и индивидуальных результатов | Наличие и адекватность распределения ролей в коллективе в ходе совместного решения проблем | Наблюдение<br>Беседа<br>Эксперимент |
| | | Сравнение коллективного и суммы личных результатов | |
| | Изменения круга общения ребенка | Рост количества друзей среди членов объединения | Социометрия<br>Анкетирование<br>Наблюдение<br>Эксперимент |
| | | Исчезновение барьеров общения по разным признакам | |
| Устойчивый интерес к предмету и к внепрограммному материалу | Место учебного предмета в жизни ребенка | Длительность и частота (интенсивность) занятий программированием вне школы «в свое удовольствие» | Беседа с родителями<br>Наблюдение |
| | Обращение к педагогу по вопросам содержания, непосредственно не связанным с изучаемым материалом | Количество обращений | Статистика (беседы при личной встрече, по телефону, e-mail) |
| | | Характер вопросов и сообщений, глубина заинтересованности | |
| Способность самостоятельно изучать материал | Наличие умения самостоятельно изучать трудные или значительные по объему темы | Степень самостоятельности (участие педагога) | Самоанализ<br>Беседа<br>Проверка работ |
| | | Качество усвоения | |

| Ожидаемый результат | Параметры | Критерии | Методы отслеживания |
|---|---|---|---|
| Умение планировать свою деятельность | Развитие навыков планирования | Количество усвоенных компонент (построение сложных планов, учет взаимосвязей при «распараллеливании работы) | Наблюдение<br>Эксперимент<br>Беседа с родителями |
| | Умение распределять нагрузку по времени | Степень равномерности распределения нагрузки | |
| Способность к самоконтролю | Умение контролировать ход выполнения работ, требующих длительного времени | Эффективность и результативность контроля | Наблюдение<br>Эксперимент<br>Беседа с родителями |
| Умение составлять олимпиадные задачи в области программирования | Успешность ребенка как автора | Уровень сложности задач | Беседа |
| | | Количество задач в год | |
| | | Красота идей | |
| Получение некоторыми школьниками научных результатов | Успешность исследовательской деятельности | Спонтанность | Наблюдение<br>Беседа<br>Отчеты детей<br>чтение, анализ |
| | | Результативность | |
| | | Широта областей исследования | |
| | | Глубина исследования | |
| | Самостоятельность при получении результатов | Степень участия руководителя | Оценка эксперта<br>Беседа с ребенком и руководителем |
| | Новизна результатов | Наличие опубликованных работ с теми же результатами у других авторов: если «да» - то степень известности результатов для школьника | Переписка<br>Работа с источниками |
| | Научная значимость результатов | Представляет ли интерес в научных кругах | Переписка |
| | Массовость | Количество школьников, занимающихся научной деятельностью | Анализ информации от детей, из школ |
| Успешное выступление школьников на соревнованиях | Рост успехов школьников (каждого в отдельности) и статистика по учебной группе | Сравнение уровня соревнований, набранных балов, дипломов, мест | Анализ результатов соревнований |
| Поступление школьников на специальности ведущих ВУЗов страны | Наличие высокого процента школьников, поступивших на соответствующие специальности ведущих ВУЗов страны | Статистика по ВУЗам | Анализ достаточно разрозненных сведений из бесед с детьми, их родителями и учителями |
| | | Статистика по профилю обучения | |

| Ожидаемый результат | Параметры | Критерии | Методы отслеживания |
|---|---|---|---|
| | Наличие учеников, для которых программирование стало профессией | Да/нет, если «да» то список | |
| Усвоение содержания программы | Глубина усвоения знаний | % материала, который ребенок запомнил | Эксперимент (проверочная работа) |
| | | | Беседа |
| | Широта применения знаний | Количество и значимость параметров задачи, при изменении которых школьник умеет ее решать | Эксперимент (проверочная работа) |
| | | | Беседа |
| Наличие определенной культуры при решении задач и работе в цифровой среде | Умение понятно излагать свои мысли как устно, так и письменно | Отсутствие неверно понятых рассуждений сверстниками и взрослыми | Наблюдение |
| | | | Сравнение результатов на соревнованиях до и после апелляции с последующим выяснение причины в беседе с ребенком |
| | | | Беседа с командами по окончании командных соревнований |
| | Отсутствие логических ошибок в рассуждениях | Расширение набора схем рассуждений, выполняемых без логических ошибок | Наблюдение |
| | | | Проверка письменных работ |
| | Умение алгоритмизировать процесс поиска решения | Увеличение числа известных школьнику алгоритмов поиска решения | Наблюдение |
| | | | Беседа |
| | | | Проверка письменных работ |
| | | Результативность применения алгоритмов поиска решения | Наблюдение |
| | | | Беседа |
| | | | Проверка письменных работ |
| Умение применять знания в смежных с программированием областях деятельности | Улучшение успеваемости, успехов на соревнованиях в смежных с программированием областях | Корреляция между успешностью занятий олимпиадным программированием и успешностью занятий математикой и естественнонаучными дисциплинами (победы в соревнованиях, успеваемость) | Анализ достаточно разрозненных сведений из бесед с детьми, их родителями и учителями |
| | | | Анализ статистических таблиц участия в соревнованиях |

**Личная карта обучающегося** (заполняется педагогом как дневник наблюдений)
Ф.И. ученика _____

| Параметры | Критерии | 1-е полу-годие | 2-е полу-годие |
|---|---|---|---|
| Изобретение способов решения проблем по красоте превосходящих авторские (общепринятые) | Статистика и красота, оригинальность таких решений | | |
| | Количество человек, отмечающих изменения, произошедшие в ребенке | | |
| Наличие обращений за помощью по предмету со стороны старших школьников и студентов | Количество обращений | | |
| Успешность выступлений на соревнованиях | Количество побед на математических соревнованиях за более старшие классы (возрастные группы) | | |
| Соотношение коллективного и индивидуальных результатов | Наличие и адекватность распределения ролей в коллективе в ходе совместного решения проблем | | |
| | Сравнение коллективного и суммы личных результатов | | |
| Изменения круга общения | Рост количества друзей среди членов творческого объединения | | |
| | Исчезновение барьеров общения по разным признакам | | |
| Место учебного предмета в жизни школьника | Длительность и частота (интенсивность) занятий программированием и математикой вне школы и объединения «в свое удовольствие» | | |
| Обращение к педагогу по вопросам содержания, непосредственно не связанным с изучаемым материалом | Количество обращений | | |
| | Характер вопросов и сообщений, глубина заинтересованности | | |
| Наличие умения самостоятельно изучать трудные или значительные по объему темы | Степень самостоятельности (участие педагога) | | |
| | Качество усвоения | | |
| Развитие навыков планирования | Количество усвоенных компонентов (построение сложных планов, учет взаимосвязей при «распараллеливании работы») | | |
| Умение распределять нагрузку по времени | Степень равномерности распределения нагрузки | | |
| Умение контролировать ход выполнения работ, требующих длительного времени | Эффективность и результативность контроля | | |
| Успешность ребенка как автора | Уровень сложности задач | | |

| Параметры | Критерии | 1-е полу-годие | 2-е полу-годие |
|---|---|---|---|
| | Количество задач в год | | |
| | Красота идей | | |
| Успешность исследовательской деятельности | Спонтанность | | |
| | Результативность | | |
| | Широта областей исследования | | |
| | Глубина исследования | | |
| Самостоятельность при получении результатов | Степень участия руководителя | | |
| Новизна результатов | Наличие опубликованных работ с теми же результатами у других авторов: если «да» - то степень известности результатов для воспитанника | | |
| Научная значимость результатов | Представляет ли интерес в научных кругах | | |
| Рост успехов ученика | Сравнение уровня соревнований, набранных балов, дипломов, мест | | |
| Улучшение успеваемости по информатике, математике | Изменения в текущей, срезовой и итоговой успеваемости | | |
| Глубина усвоения знаний | % материала, который ребенок запомнил | | |
| Широта применения знаний | Количество и значимость параметров задачи, при изменении которых школьник умеет ее решать | | |
| Умение понятно излагать свои мысли как устно, так и письменно | Отсутствие неверно понятых рассуждений сверстниками и взрослыми | | |
| Отсутствие логических ошибок в рассуждениях | Расширение набора схем рассуждений, выполняемых без логических ошибок | | |
| Умение алгоритмизировать процесс поиска решения | Увеличение числа известных школьнику алгоритмов поиска решения | | |
| | Результативность применения алгоритмов поиска решения | | |
| Улучшение успеваемости, успехов на соревнованиях в смежных областях | Корелляция между успешностью занятий программированием и успешностью занятий математикой и естественнонаучными дисциплинами (победы в соревнованиях, успеваемость) | | |

**Календарь соревнований** школьников по информатике и программированию

| **Местная компонента** *(включая городской, региональный уровень и всероссийские соревнования, проходящие в Вологодской области)* | |
|---|---|
| Школьный этап Всероссийской олимпиады школьников по информатике | октябрь |
| Муниципальный этап Всероссийской олимпиады школьников по информатике | ноябрь – декабрь |
| Региональный этап Всероссийской олимпиады школьников по информатике | январь |
| Зимняя олимпиада по программированию | январь |
| Северный математический турнир | февраль – март |
| Межрегиональная олимпиада по программированию | апрель |
| Городской слёт «Интеллект» на базе загородного лагеря «Единство» (отбор городских команд) | июнь – август |
| **Всероссийская компонента** | |
| Всероссийская командная олимпиада школьников по информатике и программированию. Отборочный (в Центральном регионе России) и финальный тур | октябрь, декабрь |
| Индивидуальная олимпиада школьников по информатике и программированию | март |
| Заключительный этап Всероссийской олимпиады школьников по информатике | март – апрель |
| Всероссийская олимпиада школьников по информатике им. Мстислава Келдыша | июнь |
| Открытая командная олимпиада по программированию в рамках фестиваля «IT-Архангельск» | декабрь |
| Турнир по программированию #DEVELOBEAR | ноябрь |
| Всероссийский игровой конкурс «КИТ – компьютеры, информатика, технологии» | ноябрь – декабрь |
| Онлайн-олимпиады и отборочные этапы на сайте Codeforces | весь год |
| Интернет-олимпиады по информатике и программированию | октябрь – март |
| Различные олимпиады по информатике из «перечня олимпиад школьников и их уровней» | март – апрель |

**Примерные образцы заданий для промежуточной аттестации и итогового контроля:**

**Задание 1.** *Однажды Васе понадобилось найти вторую порядковую статистику последовательности целых чисел, то есть такое значение, которое попадет на второе место после сортировки всех различных элементов данной последовательности. Другими словами, надо найти наименьший элемент строго больший минимума. Помогите Васе справиться с этой задачей.*

Входные данные

В первой строке входных данных содержится целое число n ($1 \le n \le 100$) — количество чисел в последовательности.

Во второй строке через пробел записаны n целых чисел — элементы последовательности. Эти числа не превосходят по модулю 100.

Выходные данные

Если в заданной последовательности вторая порядковая статистика существует, выведите ее, иначе выведите NO.

Примеры

входные данные

```
4
1 2 2 -4
```

выходные данные

```
1
```

входные данные

```
5
1 2 3 1 1
```

выходные данные

```
2
```

**Задание 2.** *На столе в ряд выложены n камней, каждый из которых может быть красного, зеленого или синего цвета. Посчитайте, какое минимальное количество камней нужно убрать со стола, чтобы любые два соседних камня имели разные цвета. Камни в ряду считаются соседними, если между ними нет других камней.*

Входные данные

В первой строке задано целое число n ($1 \le n \le 50$) — количество камней на столе.

В следующей строке задана строка s, обозначающая цвета камней. Будем считать, что камни в ряду пронумерованы целыми числами от 1 до n слева направо. Тогда

i-ый символ s равен «R», если i-ый камень красного цвета, «G» — если он зеленого цвета, и «B» — если он синего цвета.

Выходные данные

Выведите единственное целое число — ответ на задачу.

Примеры

входные данные

```
3
RRG
```

выходные данные

```
1
```

входные данные

```
5
RRRRR
```

выходные данные

```
4
```

входные данные

```
4
BRBG
```

выходные данные

```
0
```

**Задание 3.** *Вася следит за баскетбольным матчем, записывая, какая команда с какой дистанции сделала очередной бросок. Вася знает, что каждый бросок оценивается либо в 2, либо в 3 очка. Бросок оценивается в 2 очка, если дистанция, с которого он выполнен, не превосходит некоторого значения в d метров, и 3, если эта дистанция больше, чем d метров, где d — некоторое неотрицательное целое число.*

*Вася хочет, чтобы разность набранных командами очков (очки первой команды минус очки второй команды) была как можно больше. Для этого он может мысленно выбирать значение d. Помогите ему добиться желаемого.*

Входные данные

В первой строке задано целое число n ($1 \le n \le 2 \cdot 105$) — количество бросков первой команды. Далее следуют n целых чисел — дистанции бросков $a_i$ ($1 \le a_i \le 2 \cdot 109$).

Затем дается число m ($1 \le m \le 2 \cdot 105$) — количество бросков второй команды. Далее m целых чисел — дистанции бросков $b_i$ ($1 \le b_i \le 2 \cdot 109$).

Выходные данные

Выведите два числа в формате a:b — возможный в условиях задачи счёт, при котором разность a - b максимальна. Если таких счетов несколько, выведите такой, в котором число a — максимально.

Примеры

входные данные

```
3
1 2 3
2
5 6
```

выходные данные

```
9:6
```

входные данные

```
5
6 7 8 9 10
5
1 2 3 4 5
```

выходные данные

```
15:10
```

**Задание 4.** *Скобочная последовательность называется правильной, если путем вставки в нее символов «+» и «1» можно получить из нее корректное математическое выражение. Например, последовательности «(())()», «()» и «((()))» — правильные, в то время как «)(», «(()» и «(()))(» — нет.*
*Вам задана строка, состоящая из символов «(» и «)». Ваша задача найти её наидлиннейшую подстроку, которая является правильной скобочной последовательностью. Также вам надо найти количество таких подстрок.*

Входные данные

В первой строке входного файла записана непустая строка, состоящая из символов «(» и «)». Её длина не превосходит 106.

Выходные данные

Выведите длину наибольшей подстроки, являющейся правильной скобочной последовательностью, и количество таких подстрок. Если искомых подстрок не существует, то выведите «0 1» в единственную строку выходных данных.

Примеры

входные данные

```
)(((())))(()())
```

выходные данные

```
6 2
```

входные данные

```
)(
```

выходные данные

```
0 1
```

## МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ПРОГРАММЫ

В реализации программы применяются кейс технологии.

Кейс технологии включают в себя различные методы, приёмы и техники обучения, связанные с решением задач прикладного содержания. Метод кейсов (case method) в переводе с английского означает метод анализа конкретных ситуаций или метод ситуационного анализа. Обучающиеся должны проанализировать ситуацию, построить модель, предложить различные варианты решения и выбрать оптимальное. Для решения поставленной задачи в общем виде метод кейсов преследует цели: определение типов данных, формирование алгоритма с использованием функций, написание программы на языке программирования, отладка и тестирование программы. Школьник должен разработать и реализовать объектную модель задачи, проанализировать задачу с точки зрения выбора структуры представления данных, сделать соответствующие выводы. Выбор структуры представления данных определяет эффективность используемых алгоритмов, поэтому первоначальной задачей является теоретико-практическое изучение структур данных.

В обучении в рамках программы используется метод конкретных ситуаций, основанный на решении конкретных задач, просто сформулированных и легко адаптируемых под реальные ситуации или иные учебные задачи. В нашем случае это является трактовкой кейс-метода для математики и программирования. Каждый кейс в первую очередь направлен на овладение конкретной, практически полезной идеей, основной упор при общении с обучающимся должен делаться на анализ данной конкретной ситуации, а не на подбор способа её преодоления, так что код на языке программирования C++ является лишь примерным и ученик по итогам разбора и анализа должен уметь самостоятельно реализовать решить ситуацию на языке программирования C++. Тем не менее к готовому коду по возможности прикреплены комментарии, чтобы ученик, во-первых, не "терялся" в технической части языка программирования, а во-вторых, мог бы посмотреть как поставленная конкретная задача на пути разрешения кейса может быть реализована на языке программирования *(см. Образцы учебных кейсов, стр. 41)*

**О задачах олимпиадного программирования**

В отличии от обычных программ, создаваемых программистами повседневно, класс олимпиадных задач достаточно узок, но практичен с точки критериев выявления способности участников программировать за короткий срок. Как правило, олимпиадная задача представляет собой некоторую проблему, для решения которой требуется использовать свой IQ почти на пределе, однако, сам текст программы может быть совсем незначительным и помещаться на одной

странице.

**Условная классификация олимпиадных задач**

— Конструктив — задачи, обычно не требующие каких-либо знаний в области олимпиадных алгоритмов. Ответ на задачу строится последовательно согласно видению автора кода.

— Арифметика — математические задачи (теория чисел, комбинаторика), работа с большими числами (длинная арифметика). Такие задачи, как правило, требуют знания формул, умение их применять, а код программ обычно бывает небольшим.

— Геометрия — геометрические задачи, здесь может быть описана какая-либо ситуация взаимодействия тел на плоскости и в пространстве.

— Динамическое программирование — задачи, направленные на выявление рекуррентных соотношений.

— Сортировка, поиск и последовательности — работа с данными, представленными в виде массива.

— Структуры данных — построение (использование) различных структур данных для эффективного решения задач.

— Графы — задачи с графами (структурами данных, основанных на вершинах и ребрах).

— Рекурсия — задачи на поиск с рекурсивным перебором вариантов.

Задачи могут сочетать в себе сразу несколько направлений и часто бывает сложно конкретную задачу отнести к тому или иному разделу. Любая олимпиадная задача подразумевает входные и выходные данные. Т.е. в формулировке задания обязательным образом описан формат входных и выходных данных, а Ваша программа должна считать эти данные, обработать и вывести результат в установленном формате. Чаще всего чтение происходит из некоторого файла (например, INPUT.TXT), а вывод в некоторый файл (например, OUTPUT.TXT). Т.е. для решения олимпиадных задач нужно уметь работать с файлами: читать, создавать и писать в них. Стоит заметить, что многие системы, например http://acm.timus.ru, используют консольный режим ввода-вывода и работа с файлами в них не приветствуется. Помимо условия задачи, правил ввода и вывода информации на каждую задачу накладываются ограничения на время выполнения и используемую программой оперативную память.

**Правила проведения олимпиад по программированию**

Соревнования обычно длятся 4,5–5 часов. На это время участнику предоставляется компьютер и предлагается решить несколько задач. Набор задач

для всех участников одинаковый. Во время тура запрещается пользоваться личными компьютерами, калькуляторами, электронными записными книжками, средствами связи (пейджерами, телефонами), а также учебной литературой и личными записями. Рабочими языками, как правило, считаются Паскаль и Си. На рабочем месте участника устанавливаются полностью системы BorlandPascal 7.0 и BorlandC/C++ 3.1. В последнее время все чаще на соревнованиях используется FreePascal. Участник сам определяет язык и систему, в которой будет работать. Разные задачи можно решать с использованием разных языков программирования.

В процессе работы участники создают программы и отправляют их жюри для проверки. Жюри проводит тестирование решения на некотором наборе тестов и сообщает автору программы результаты тестирования. Решение, прошедшее все тесты, принимается, в противном случае программа возвращается на доработку. Побеждает участник, решивший наибольшее количество задач с наименьшими затратами времени. Главное, на что нужно обратить внимание потенциальных участников соревнований, это тот факт, что чаще всего проверка решений проводится в автоматическом режиме. Жюри автоматически компилирует программу и автоматически запускает ее на тестовых примерах. Поэтому для участника чрезвычайно важно выполнить все требования, предъявляемые к решению, т. к. как бы ни был хорош и эффективен алгоритм решения задачи, простейшая синтаксическая ошибка, полученная при компиляции, испортит все.

Отправлять жюри следует исходный текст на любом из допустимых языков программирования. Причем программа должна быть реализована в виде одного файла и не требовать подключения каких-либо модулей. Часто использование даже фразы “usescrt” может привести к ошибке.

Все входные данные вводятся из файла, указанного в условии задачи (часто это файл “INPUT.TXT”), результат записывается в выходной файл (например, “OUTPUT.TXT”). Входные и выходные файлы размещаются в текущем каталоге DOS. Типичная ошибка новичка – попытка указать пути к файлам, например, написать ’a:\input.txt’. Поскольку у жюри в дисководе диск отсутствует, то программа выдает сообщение об ошибке и завершает работу.

Необходимо строго соблюдать форматы входных и выходных данных, указанных в условии задачи. Никаких лишних символов в выходном файле быть не должно.

Программы не должны выводить что-либо на экран или ожидать ввода данных с клавиатуры. Очень часто команда получает от жюри сообщение о том, что при проведении тестирования превышен лимит времени на выполнение теста. И бывает, что «зависание» программы происходит благодаря оператору “readln”, забытому в конце текста.

Все необходимые директивы компиляции следует размещать внутри исходных текстов.

При решении задачи нельзя использовать чтение и запись векторов прерываний, защищенный режим, работу с другими файлами, кроме явно указанных в условии. Нарушение данного требования рассматривается как попытка обмануть жюри и приводит к дисквалификации участника.

И последнее. Так как все решения жюри окончательные и обжалованию не подлежат, то рекомендуется соблюдать правила вежливости при любом общении с жюри.

Правила конкретной олимпиады могут несколько отличаться от приведенных выше, но до начала соревнований вас обязательно познакомят с действующими правилами и используемым программным обеспечением.

**Система мотивирования обучающихся к активной деятельности**

— Рейтинговая система оценки достижений.

— Нетрадиционные формы проведения занятий (олимпиады, турниры, бои и т.п.).

— Возможности подготовки поступления в ВУЗ, сдачи ЕГЭ, профориентации.

— Система поощрений (грамоты, дипломы, участие в слете, турнирах, пополнение портфолио и др.).

— В течение года обучающиеся творческого объединения принимают участие в соревнованиях разного уровня.

## ОБРАЗЦЫ УЧЕБНЫХ КЕЙСОВ

**Кейс-задача 1.** *Имеется шахматная доска размером N×M. Сколько на ней чёрных клеток?*

Разбор и решение: Мы должны вспомнить, что клетка A1 традиционно чёрная. Тогда если всего клеток нечётное число, то чёрных будет на 1 больше (Представим, что мы решили покрасить доску самостоятельно и красить решили «змейкой». Две соседние клетки на шахматной доске разноцветны, т. е. мы красим чёрная-белая-чёрная-…, начиная с чёрной клетки). В свою очередь для чётного числа клеток двух цветов будет поровну.

Доска задаётся двумя параметрами — длиной и шириной. Для их считывания и хранения понадобится создать две переменные целочисленного типа. Количество клеток считается по формуле (длина × ширина).

Первый способ решения — заметить, что по сути кол-во чёрных клеток является половиной от общего числа с округлением вверх. Тогда ответ можно посчитать одной лишь формулой.

Код решения формулой:

```
#include <iostream>

using namespace std;

int main() {
    int length,_width;   // создали переменные для длины и ширины
    cin >> length >> width; // узнали значения длины и ширины, сохранили в переменные
    cout << (length * width + 1) / 2;        // вывели ответ на задачу
    return 0;
}
```

Разбор и решение вторым способом: Во втором способе мы вспомним, что произведение двух чисел (а общее число клеток является произведением двух чисел, как мы уже выяснили в первом решении) является нечётным тогда и только тогда, когда оба числа сами являются нечётными.

С помощью такой проверки обоих чисел на нечётность и реализуем ветвление на случаи.

Код решения условным оператором:

```
#include <iostream>
```

```cpp
using namespace std;

int main() {
    int length, width;   // создали переменные для длины и ширины
    cin >> length >> width;   // узнали значения длины и ширины, сохранили в переменные
    if (length % 2 != 0 && width % 2 != 0) { // если клеток нечётное число
    // вспомним, что && – это одновременная верность двух условий
        cout << length * width / 2 + 1;     // на одну чёрную клетку больше
    } else {                                // иначе
        cout << length * width / 2;         // клеток двух цветов поровну
    }    // ответ выведется в нужной (под конкретный случай) ветке условного оператора
    return 0;
}
```

**Кейс-задача 2.** *Даны массы 4 гирек. Нужно расставить их в порядке возрастания (неубывания) масс.*

Разбор и решение: Для решения воспользуемся полезной функцией. swap(a, b) меняет местами значения в переменных a и b. После её применения в переменной a будет храниться то значение, которое раньше хранилось в переменной b. И наоборот.

Вспомним также, как объявляются простейшие массивы — фиксированного размера.

Сортировать будем следующим способом: попарно будем сравнивать гирьки; теперь если видим проблему, то будем её разрешать (переставлять пару гирек в правильном порядке).

Проблема для нас — случай, когда гирька с меньшим номером имеет большую массу, чем гирька с большим номером.

Код решения:

```cpp
#include <iostream>
#include <algorithm> // новая для нас библиотека algorithm
                     // в ней описаны полезные функции, в том числе и swap()
using namespace std;

int main() {
    int num[4];     // создаём массив с 4 ячейками. Размер массива указывается ЧИСЛОМ
    cin >> num[0] >> num[1] >> num[2] >> num[3];
// сохранили значения 4 чисел в ячейки массива
// обратите внимание, несмотря на то, что размер = 4, ячейки нумеруются с 0!!
// значит, последняя имеет номер равный числу (размер - 1). Здесь последняя имеет номер 3!!
```

```
    // внутри if'ов будем писать те условия, которые говорят о проблеме,
    // которую мы и будем разрешать с помощью swap – перемены местами
    if (num[0] > num[1]) {    // если значение нулевой ячейки больше, чем первой
        swap(num[0], num[1]);     // меняем их значения местами
    }
    if (num[0] > num[2]) {
        swap(num[0], num[2]);
    }
    if (num[0] > num[3]) {
        swap(num[0], num[3]);
    }
// За первые 3 if'а мы сравнили нулевую ячейку со всеми другими, значит, после этих 3 if'ов
// в нулевой ячейке будет лежать наименьшее число. Больше нулевую ячейку не трогаем

    if (num[1] > num[2]) {
        swap(num[1], num[2]);
    }
    if (num[1] > num[3]) {
        swap(num[1], num[3]);
    }
// За эти 2 if'а мы положили второе по размеру (второе наименьшее) число в num[1]
// Первую (номер 1) ячейку больше тоже не трогаем

    if (num[2] > num[3]) {
        swap(num[2], num[3]);
    }    // теперь в num[2] предпоследнее по значению, а в num[3] наибольшее
    cout << num[0] << ' ' << num[1] << ' ' << num[2] << ' ' << num[3];
    return 0;
}
```

**Кейс-задача 3.** *Дано число N. Затем дано N чисел. Посчитайте их сумму.*

Разбор и решение: Для решения данной задачи задумайтесь, каким образом вы суммируете у себя в голове числа, читая их слева направо.

Это один из простейших алгоритмов — однопроходный алгоритм. Т.е. для решения данной задачи нам не потребуется запоминать эти N чисел надолго. Достаточно взглянуть на каждое из них однажды и добавить в «общую копилку».

Код решения:

```
#include <iostream>
```

```
using namespace std;

int main() {
    int n;          // создали переменную
    cin >> n;       // сохранили в неё количество чисел
    int sum = 0;    // создали переменную, где будет храниться сумма чисел
    // пока мы не знаем ни одно из n чисел, значение у этой переменной = 0
    for (int i = 0; i < n; ++i) { // создаём цикл на n шагов, т.к. у нас n чисел
// данный цикл будет иметь n шагов, т.к. чисел от 0 до n, не включая n, ровно n штук
        int cur;        // переменная, где будет храниться текущее из n чисел
        cin >> cur;     // узнаём значение очередного числа
        sum += cur;     // sum = sum + cur  (к сумме прибавляется данное число)
    }
    cout << sum;    // в качестве ответа пишем сумму после всех n шагов
    return 0;
}
```

**Кейс-задача 4.** *Дано число N. Затем дано N чисел. Найдите наибольшее из них.*

Разбор и решение: Ещё один однопроходный алгоритм.

Начнём с того же: задумайтесь, как бы вы, читая числа слева направо, пытались бы помнить, какое из них на данный момент являлось наибольшим. Вот так постепенно и будем находить локальный максимум среди первых скольких-то чисел ввода (сначала наибольшее среди одного числа, потом — наибольшее среди первых двух, затем — первых трёх чисел и так далее). Когда мы посмотрим на каждое из них, мы найдём максимум среди первых N чисел ввода, т. е. попросту максимум среди всех.

Код решения:

```
#include <iostream>

using namespace std;

int main() {
    int n;          // количество чисел
    cin >> n;

    int maximum = 0;     // переменная, где будет храниться максимум
    // можно взять в качестве начального значения 0, т.к. числа натуральные
    // и поэтому любое из них превышает 0
```

```
    for (int i = 0; i < n; ++i) {  // цикл на n шагов
// После x шагов цикла у нас в переменной maximum будет храниться наибольшее среди
// первых x чисел. Соответственно, после всех n шагов будет известен общий максимум
        int cur;        // текущее число
        cin >> cur;

        if (cur > maximum) { // если текущее превысило максимум, то оно
            maximum = cur; // теперь новый (на момент текущего шага) максимум
        }
    }
    cout << maximum;     // выводим наибольшее из n чисел
    return 0;
}
```

**Кейс-задача 5.** *Вам дано число. Нужно написать всего его делители на экран в порядке возрастания.*

Разбор и решение: Те из вас, кто хорошо изучал математику, должны знать, что все делители числа можно разбить на пары, которые в произведении дают само число. Например, для числа 12:

$$12 = 1 \times 12; \qquad 12 = 2 \times 6; \qquad 12 = 3 \times 4.$$

Для числа, являющегося квадратом некоторого натурального числа x, т. е. для числа $x^2$, можно условно считать, что число x (а это корень исходного числа и одновременно его делитель), вступает в пару с самим собой. Например, для числа 36:

$$36 = 1 \times 36; 36 = 2 \times 18; 36 = 3 \times 12; \; 36 = 4 \times 9; \; \underline{36 = 6 \times 6}$$

Несложно заметить, что меньший из подобной пары делителей некоторого числа y всегда не больше $\sqrt{y}$, а его напарник — не меньше, и перебирать делители достаточно до корня из числа, добавляя делители парами.

У подобного алгоритма есть для нас минус: числа добавляются парами и получается не отсортированный (не расставленный по возрастанию) порядок. То есть делители придётся добавлять в массив, который затем нужно будет отсортировать.

Поэтому здесь мы напишем менее эффективный, но более простой и не требующий сохранения делителей в массив, алгоритм.

Код решения:

```
#include <iostream>
```

```
using namespace std;

int main() {
     int number;     // переменная для исходного числа
     cin >> number;
     for (int d = 1; d <= number; ++d) {
     // В качестве счётчика цикла будем использовать потенциальных кандидатов
     // на роль делителя числа number. Натуральные делители есть смысл искать
     // только среди чисел от 1 до number'а.
          if (number % d == 0) {    // если d является делителем number'а
               cout << d << ' ';    // выводим его на экран
          }                         // (делители окажутся выведены через пробел)
     }
     return 0;
}
```

**Кейс-задача 6.** *Дано натуральное число N. Напишите первое число Фибоначчи, большее данного N.*

Разбор и решение: Числа Фибоначчи — такая бесконечная последовательность, где каждое число равно сумме двух предыдущих в последовательности. Первые два числа последовательности невозможно определить данной зависимостью, так что будем считать, что они равны 0 и 1 соответственно. Тогда данную последовательность можно определить формулами:

$F_0 = 0;$ $F_1 = 1;$ $F_n = F_{n-1} + F_{n-2}$, где $n \geq 2$

Воспользуемся этими формулами и будем считать очередное число Фибоначчи через два предыдущих до тех пор, пока оно не превысит заданное число N.

Заметьте, *«до тех пор, пока»*. Т.е. у нас имеется условие для продолжения повторений (шагов цикла). Для таких ситуаций мы обычно используем цикл **while**. А одним шагом цикла здесь будет подсчёт очередного числа Фибоначчи через два предыдущих.

Код решения:

```
#include <iostream>

using namespace std;

int main() {
```

```
    int n;     // переменная для исходного числа
    cin >> n;

    int previous = 0, current = 1;
    // переменные для предыдущего и текущего чисел Фибоначчи
    while (current <= n) {
        int next = previous + current; // временная переменная для нового числа Фиб.

        previous = current;  // теперь текущее становится предыдущим
        current = next;      // а новое – текущим
    }
    cout << current;     // выводим то значение, которое было посчитано на последнем шаге
    // цикла. После цикла в current будет лежать первое число Фибоначчи,  не удовлетворяющее
    // условию при while, т. е. первое число Фибоначчи, удовлетворяющее обратному условию,
    // т. е. первое, строго большее, чем n. Что и требовалось по условию задачи.
    return 0;
}
```

**Кейс-задача 7.** *Класс Саши на уроках математики в последнее время изучал признаки делимости. Сам Саша, к сожалению, всё это время проболел. Теперь, получив на руки домашнее задание, он даже не знает как понять, делится ли то или иное число на 3. Зато его друг Серёжа обещал помочь с ответами, если Саша напишет ему суммы цифр всех чисел из домашнего задания. Саша по природе очень ленив, так что попросил вас написать для него программу, которая по числу N — кол-ву чисел из домашнего задания — и списку этих чисел, подготовит текст для сообщения Серёже.*

Разбор и решение: Нам нужно для каждого из N чисел посчитать его сумму цифр. Для разбиения числа на цифры, мы будем пользоваться двумя «идеями-трюками».

Первый: получить последнюю цифру десятичного числа. Здесь нужно вспомнить, в чём смысл каждого разряда десятичного числа. Конкретно, разряд единиц — количество единиц ($10^0$) в нашем числе, разряд десятков — количество десятков ($10^1$), разряд сотен — количество сотен ($10^2$) и т. д., и если всю эту информацию сложить между собой (разряд1 $\times$ $10^0$ + разряд10 $\times$ $10^1$ + разряд100 $\times$ $10^2$ + ...), то и получается в точности наше число. Получается, если взять остаток от деления нашего числа на 10, информация из остальных разрядов исчезнет, т. к. при суммировании поразрядно каждый из них входил с коэффициентом, кратным 10 (10 в степени строго больше 0).

Отсюда вытекает и вторая идея-трюк. Чтобы полностью избавиться (отрезать / стереть / отсечь) от последней цифры, нужно просто поделить число на 10 целочисленно и мы получим то же самое число, но без последней своей цифры

(т. к. она единственная входила в суммирование с коэффициентом не кратным 10). Комбинируя эти два трюка между собой, мы можем узнать любую цифру нашего числа.

Как определить, что все цифры закончились и больше суммировать не нужно? Когда цифры закончатся, число станет равно 0. Если число изначально было 0, то сумма цифр данного числа сразу равна 0 и суммировать тоже не нужно.

Код решения:

```
#include <iostream>

using namespace std;

int main() {
	int n;	// переменная для количества чисел в домашнем задании
	cin >> n;

	for (int i = 0; i < n; ++i) {
	// создаём цикл на n шагов: по шагу на каждое число ввода; данный цикл
	// будет иметь n шагов, т.к. чисел от 0 до n, не включая n, ровно n штук.
		int sum = 0, number; // переменные для суммы цифр и для считывания самого числа
		cin >> number;		// узнаём значение очередного числа
		while (number != 0) {	// пока цифры не закончились
			sum += (number % 10);	// прибавляем текущую последнюю цифру к сумме
			number /= 10;	// после прибавления убираем данную последнюю цифру из числа
		}
		cout << sum << '\n'; // после подсчёта пишем сумму цифр и переводим строку
	}
	return 0;
}
```

**Кейс-задача 8.** *В течение января Саша каждый день ездил на автобусе от дома и домой, записывая в текстовый документ номера билетов (в автобусах Сашиного города используются билеты с номерами из 6 десятичных цифр). Январь закончился, и ему стало интересно, сколько раз за месяц Саше достался счастливый билет. Помогите ему получить ответ на данный вопрос.*

Разбор и решение (разбиение на цифры): Во-первых, мы помним, что в январе 31 день, а Саша каждый день ездил на автобусе два раза, т. е. билетов в списке всего 62 штуки. Во-вторых, так как для определения «счастья» нужно суммировать цифры в разных половинах шестизначного билета, мы для каждого билета

сначала «отрежем» и просуммируем последние три цифры, а затем то же самое сделаем и для оставшихся «в живых» трёх цифр. После этого останется только сравнить посчитанные суммы в двух половинах текущего билета и перейти к обработке следующего.

Решение задачи тогда представляет из себя цикл на 62 шага, каждый из которых — это обработка одного билета. Обработка одного билета состоит из таких этапов:

1) Три раза повторяем набор операций: получить текущую последнюю цифру билета, добавить полученную цифру в «общую копилку» для суммы правой половины билета, избавить билет от текущей его последней цифры;

2) Три раза повторяем набор операций: получить текущую последнюю цифру билета, добавить полученную цифру в «общую копилку» для суммы левой половины билета, избавить билет от текущей его последней цифры;

3) Сравниваем посчитанные суммы в левой и правой половине билета. Если они равны, то добавляем к ответу единицу.

Код решения №1:

```
#include <iostream>

using namespace std;

int main() {
    int nLucky = 0; // переменная для ответа на задачу – кол-ва счастливых билетов
    for (int k = 0; k < 62; ++k) {
        int ticket;    // переменная для считывания текущего билета
        cin >> ticket;

        int right = 0;
        // переменная, где будет храниться сумма цифр правой половины билета
        for (int i = 0; i < 3; ++i) {
            right += ticket % 10;     // добавляем текущую последнюю цифру
                                      // к сумме правой половины
            ticket /= 10;
            // лишаем билет последней цифры, т.к. она уже обработана нами
        }

        int left = 0;
        // переменная, где будет храниться сумма цифр левой половины билета
        for (int i = 0; i < 3; ++i) {
            left += ticket % 10; // добавляем текущую последнюю цифру
                                 // к сумме левой половины
```

```
            ticket /= 10;
            // лишаем билет последней цифры, т.к. она уже обработана нами
        }

        if (left == right) { // если суммы половин равны,
            nLucky++;            // то мы встретили ещё один счастливый билет
        }
    }
    cout << nLucky;
    return 0;
}
```

Разбор и решение (символьное): Эту задачу мы уже решили одним способом (в разделе «Разбиение на цифры»). Сейчас давайте решим и вторым.

Во-первых, мы помним, что в январе 31 день, а Саша каждый день ездил на автобусе два раза, т. е. билетов в списке всего 62 штуки. Во-вторых, так как для определения «счастья» нужно суммировать цифры в разных половинах шестизначного билета, давайте сразу считывать поциферно, посимвольно, а для этого будем пользоваться переменной символьного типа данных **char**. В этом и будет особенность нашего второго способа решения.

Вспомним, что хоть и в переменных данного типа данных хранится то, что визуально является каким-то символом, хранится он в с помощью своего кода (какого-то целого числа, поставленного данному символу в соответствие).

Так как код символа-цифры не совпадает с числовым значением данной цифры (например, '0' = 48, '1' = 49 и т.д.), для получения правильной суммы цифр будем от кода каждого считанного символа (который визуально является цифрой, а считан нами в переменную типа **char**) переходить к арифметическому значению данной цифры с помощью вычитания кода символа '0'.

Чтобы это понять, вспомним, что с таблице символов все цифры хранятся последовательно, т. е. код символа '1' на 1 больше, чем код символа '0', код '2' на 2 больше кода '0' и т. д.). И с точки зрения математики получается, что:

$$('d' - '0') = d, \qquad \text{где d — это какая-то десятичная цифра.}$$

Код решения №2:

```
#include <iostream>
#include <stdio.h>
using namespace std;

int main() {
    int nLucky = 0;
```

```
    for (int k = 0; k < 62; ++k) {
        int first = 0;  // переменная, где будет храниться сумма первой половины билета
        for (int i = 0; i < 3; ++i) { // Создаем цикл на (длина_билета ÷ 2) шагов
        // (по шагу на каждую цифру половины билета). У нас длина равна шести.
        // Чисел от 0 до 3 (не включ. 3) ровно 3 штуки.
            char symb = getchar();          // считываем очередную цифру в билете
            first += (symb - '0');    // вычитаем из кода символа код '0', таким
            // образом переходим к значению цифры и добавляем его к сумме 1-й половины
        }
        int second = 0; // переменная, где будет храниться сумма второй половины билета
        for (int i = 0; i < 3; ++i) { // аналогичная длина цикла
            char symb = getchar();          // считываем очередную цифру в билете
            second += (symb - '0');   // вычитаем из кода символа код '0', таким
            // образом переходим к значению цифры и добавляем его к сумме 2-й половины
        }
        getchar();// пропускаем символ перевода (окончания) строки (ENTER,
                  // который поставлен после ввода номера очередного билета)
        if (first == second) {    // если суммы половин равны,
            nLucky++;             // то мы встретили ещё один счастливый билет
        }
    }
    cout << nLucky;// выводим ответ – кол-во счастливых билетов
    return 0;
}
```

**Кейс-задача 9.** *С клавиатуры задаётся символ. Нужно определить его «внешний вид»*

Разбор и решение: Для решения понадобится знать другой тип данных — символьный, тип данных **char**. Одновременно будет разобрана одна из функций для считывания символа.

Символы в компьютере кодируются, т. е. хранятся с помощью кодов. Таким образом рождаются так называемые «таблицы символов». При этом цифры хранятся в такой таблице последовательно от 0 до 9 (т. е. код следующей цифры на 1 больше, чем код предшествующей). Аналогично и буквы закодированы последовательно соответственно латинскому алфавиту от A до Z (т. е. код следующей в алфавите буквы на 1 больше, чем код предшествующей) и от a до z (то же самое, только для строчных букв латинского алфавита).

Код решения:

```
#include <iostream>
```

```
#include <stdio.h>  // ВНИМАНИЕ! Мы подключили новую библиотеку – stdio.h (она же cstdio)
                    // Она требуется для getchar – функции, которую мы используем ниже
using namespace std;

int main() {
     char symb = getchar();
     // getchar считывает и возвращает следующий после курсора символ во вводе
     if ('A' <= symb && symb <= 'Z') {    // если лежит в промежутке от заглавной A до Z,
          cout << "Буква";                // то буква
     } else if ('a' <= symb && symb <= 'z') { // иначе, если от строчной a до z,
          cout << "Буква";                          // то тоже буква
     } else if ('0' <= symb && symb <= '9') { // иначе, если от 0 до 9,
          cout << "Цифра";                          // то цифра
     } else {
          cout << "Знак"; // всё остальное условно будем считать знаками
     }
     return 0;
}
```

**Кейс-задача 10.** *В одну строку записаны символы. Найти количество цифр среди них.*

Разбор и решение: Главная проблема, которая должна вам бросаться в глаза, состоит в том, что нам не указано, сколько символов будет введено. Соответственно, нам нужно научиться считывать символьные данные вне зависимости от их количества при условии, что мы знаем, что они все расположены ровно в одной строке.

Здесь мы должны вспомнить, что наша функция getchar() считывает абсолютно любой символ. А среди символов существуют и служебные (такие, которые вы возможно и не видите). Например, одними из таких являются символ перевода (конца) строки и символ конца файла. Их мы и будем использовать в качестве индикатора окончания ввода, т. е. встретив какой-то из них мы поймём, что строка ввода закончена.

Задача опять же однопроходная, так как достаточно ровно один раз посмотреть на символ, чтобы понять, является он цифрой или нет, и запомнить данный факт. Как только про текущий символ всё понятно, мы приступаем к обработке следующего. И так пока строка не закончится.

Код решения:

```
#include <iostream>
#include <stdio.h>
```

```
using namespace std;

int main() {
	char symb = getchar();		// переменная типа данных char
	// getchar считывает и возвращает следующий после курсора символ во вводе
	// сейчас мы считали первый символ введенной строки

	int digits = 0; // количество цифр во вводе изначально равно 0
	while (symb != '\n' && symb != EOF) {
		// пока мы не встретили конец строки или конец текста
		// '\n' – служебный символ конца строки (перевод строки)
		// EOF – служебный символ конца файла (текста)
		if ('0' <= symb && symb <= '9') {   // если символ является цифрой
			digits++;		// увеличиваем счётчик числа цифр на 1
		}
		symb = getchar();	// когда с текущим символом сработано, читаем следующий
	}
	cout << digits;		// выводим ответ на задачу
	return 0;
}
```

**Кейс-задача 11.** *В первой строке входных данных дано число N. Во второй введено N целых чисел. Выведите их на экран в обратном порядке (сначала последнее, затем предпоследнее и т. д.)*

Разбор и решение: Для решения задачи нам требуется знание способов хранения совокупностей данных (например, массив). Массив — совокупность (набор) однотипных данных, приведённых в одну систему (чаще всего элементы в памяти хранятся непосредственно друг за другом и доступ к ним осуществляется с помощью индексов (номеров)). В качестве шаблона для реализации массива мы воспользуемся контейнером vector.

Почему нам потребуются массивы? Заметим, что мы можем начать выводить элементы только к моменту, когда последний из них будет считан (т. к. именно с него начинается вывод данных). Но затем нам понадобится выводить на экран и все остальные элементы, т. е. их значения (и порядок) должны быть где-то сохранены. Поскольку все элементы однотипные (целые числа), массив для реализации хранения и обращения к элементам подходит хорошо. Массив будет иметь размер N. Массивы в C++ нумеруются с ноля, т. е. самый начальный элемент имеет номер 0. Получается, что последний элемент будет иметь номер (N-1) — проверьте этот факт, если сходу не понимаете, почему.

Код решения:

```
#include <iostream>
#include <vector>    // подключили библиотеку для использования контейнера vector
using namespace std;

int main() {
     int N;     // количество чисел
     cin >> N;
     vector<int> mas(N); // реализуем массив (вектор) целых чисел размером N ячеек
     for (int i = 0; i < N; ++i) {
     // мы будем считывать наши числа в элементы массива
     // первый элемент будет иметь номер 0, последний – (N-1)
          cin >> mas[i];
          // на i-м шаге цикла считываем очередной число в i-ую ячейку массива
     }
     for (int i = N - 1; i >= 0; --i) {
     // т. к. числа нам нужно вывести в обратном порядке,
     // начинаем работу с (N-1)-го (последнего) элемента, а заканчиваем нулевым.
          cout << mas[i] << " ";
     }
     return 0;
}
```

**Кейс-задача 12.** *Гимназия города Энска решила провести вечер встречи выпускников. Для этого на сайте гимназии была открыта электронная запись, где от будущих участников встречи требовалось только указать год своего выпуска. Гимназия собрала данные и просит теперь вас определить, сколько нужно выделить аудиторий, чтобы каждой параллели досталась отдельная аудитория.*

Разбор и решение: По сути нам достаточно определить, сколько различных годов выпуска нам было введено — сколько разновидностей года выпуска будет, столько различных встреч и произойдёт. Условно будем считать, что годов выпуска более ранних, чем 1900 не будет.

Идеей нашего решения будет создание ассоциативного массива под названием needs . В качестве индекса такого массива будем использовать год выпуска, а в качестве значений ячейки — встречался ли такой год среди заявок или нет. В целях экономии памяти под данный массив будем считать, что 1900 году соответствует ячейка номер 0, 1901 — ячейка номер 1, …, 2019 году —

ячейка номер 119. Т. е. из года достаточно вычесть 1900, чтобы получить номер соответствующей ячейки.

Пусть изначально в массиве для всех годов выпуска хранится false (ложь) — показатель, что пока что на данный год не было заявок. Как только мы узнали, что заявка на год выпуска Y имеется, мы ячейке под номером (Y – 1900) зададим значение true (правда).

Тогда после обработки всех заявок останется только лишь посчитать количество true'шных ячеек.

Код решения:

```
#include <iostream>
#include <vector>    // подключили библиотеку для использования контейнера vector
using namespace std;

int main() {
	int k;    // количество заявок
	cin >> k;
	vector<char> needs(120, false);// с помощью контейнера вектор реализуем асс. массив
	for (int i = 0; i < k; ++i) {  // цикл, для обработки заявок
		int year;       // переменная, куда мы считаем год из очередной заявки
		cin >> year;
		needs[year - 1900] = true;
	}
	int halls = 0;       // переменная для количества нужных аудиторий
	for (int i = 0; i < 120; ++i) {
		if (needs[i]) {      // если заявки на данный год выпуска поступили,
			halls++;        // выделяем под данную параллель аудиторию
		}
	}
	cout << halls;
	return 0;
}
```

## ЛИТЕРАТУРА ДЛЯ ПЕДАГОГОВ

1. Еремин А.С. Кейс-метод: наиболее распространенная форма реализации компетентностного подхода // Инновации в образовании. – 2010. – №2. – С.61-81.
2. Конова Е.А., Поллак Г.А. Интерактивный метод обучения программированию с использованием технологии кейс-стади // Информатика и образование. – 2013. – №8. – С.25.
3. Конова Е.А., Поллак Г.А. Обучение программированию с использованием метода кейсов // Специализированное профессиональное издание открытого доступа «Образование 3000». – 2014. – №2. – С.57-63.

## ЛИТЕРАТУРА ДЛЯ ОБУЧАЮЩИХСЯ

1. Абрамов С.А. Математические построения и программирование.— М.: Наука, 1978.
2. Аммерал Л. Принципы программирования в машинной графике.— М.: Сол Систем, 1992.
3. Бочков С.О., Субботин Д.М. Язык программирования Си для персонального компьютера.—М.: СП Диалог, 1991.
4. Брудно А.Л. Каплан Л.И. Московские олимпиады по программированию.— М.: Наука, 1990.
5. Вирт Н. Алгоритмы и структуры данных.— М.: Мир, 1991.
6. Керниган Б., Ритчи Д. Язык программирования Си.— М.: Финансы и статистика, 1992.
7. Шень А. Программирование: теоремы и задачи. — М.: МЦНМО, 1995.
8. Кнут Д. Искусство программирования для ЭВМ.— М.: Мир, 1976.- Т.1.
9. Колмогоров А.Н., Фомин С.В. Элементы теории функций и функционального анализа.— М.: Наука, 1989.
10. Островский С.Л. Алгоритм Брезенхема // Информатика.— 1996.- № 32.
11. Пильщиков В.Н. Сборник упражнений по языку Паскаль.— М.: Наука, 1989.
12. Шикин Е.В. Боресков А.В. Компьютерная графика.- М.: Диалог- МИФИ, 1995.
13. Задачи по программированию/ С. А. Абрамов, Г. Г. Гнездилова, Е. Н. Капустина, М. И. Селюн.- М.: Наука, 1988.
14. Касаткин В.Н. Информация. Алгоритмы. ЭВМ.— М.: Просвещение, 1991. Программирование: вводный курс / Под ред. Д. Школьника.— М.: МЦНМО, 1995.
15. Круглински, Уингоу, Шеферд. Программирование на Microsoft Visual C++ 6.0 для профессионалов (PDF[in RAR], 54.4 МБ) C++
16. ANSI. C++ International Standard (second edition, 2003-10-15) (PDF, 2.3 МБ)
17. Эккель. Философия C++. Введение в стандартный C++ (2-е изд.) (DJVU, 6.4 МБ)

18. Эккель, Эллисон. Философия C++. Практическое программирование (DJVU, 6.5 МБ)
19. Саттер. Решение сложных задач на C++. 87 головоломных примеров с решениями (DJVU, 3.8 МБ)
20. Саттер. Новые сложные задачи на C++. 40 новых головоломных примеров с решениями (DJVU, 3.6 МБ)
21. Страуструп. Язык программирования C++ (2-е, дополненное изд.) (PDF, 2.9 МБ)
22. Stroustrup. The C++ Programming Language (3rd edition) (PDF, 3.4 МБ)
23. Abrahams, Gurtovoy. C++ Template Metaprogramming: Concepts, Tools, and Techniques
24. from Boost and Beyond (CHM, 0.62 МБ)
25. Джосьютис. C++. Стандартная библиотека. Для профессионалов (DJVU, 4.8 МБ)
26. Джосьютис. C++. Стандартная библиотека. Для профессионалов (CD к книге) (ZIP, 0.14 МБ)
27. Vandervoorde, Josuttis. C++ Templates: The Complete Guide (CHM, 0.72 МБ)
28. Sutter, Alexandrescu. C++ Coding Standards: 101 Rules, Guidelines, and Best Practices (CHM, 0.51 МБ)
29. Голуб. Веревка достаточной длины, чтобы выстрелить себе в ногу.
30. Правила программирования на C и C++ (PDF, 1.29 МБ)
31. Meyers. Effective C++. More effective C++ (CHM, 2.0 МБ)
32. Дьюхэрст. Скользкие места C++. Как избежать проблем при проектировании и компиляции ваших программ (DJVU, 9.3 МБ)
33. Дьюхэрст. C++. Священные знания (DJVU, 6.7 МБ)
34. Кормен, Лейзерсон, Ривест, Штайн. Алгоритмы. Построение и анализ (2-е зд.)

## КАЛЕНДАРНЫЙ УЧЕБНЫЙ ГРАФИК

**Название программы:** «Фундаментальные алгоритмы»
**ФИО педагога:** _____
**Учебный год:** 2022/2023
**Продолжительность реализации программы:** 9 месяцев
**Количество часов:** 144 часа
**Группа №** _____
**Расписание занятий:** _____
**Праздничные дни:** 4 ноября; 31 декабря; 1, 2, 3, 4, 5, 6, 7, 8 января; 23 февраля; 8 марта; 1, 9, мая
**Промежуточная аттестация:** декабрь
**Итоговый контроль:** май

| Дата | Тема / Тематический блок | Часов теор. | Часов практ | Итого часов |
|---|---|---|---|---|
| **I** | **Геометрия в программировании** | **3** | **5** | **8** |
| | Основные объекты: точка, отрезок, луч, угол, многоугольник, окружность и прочие. Геометрия в координатах. Идея симметрии (центральная, осевая), параллельного переноса, вращения. Взаимные расположения точек и прямых | | 2 | |
| | Геометрия векторов. Скалярное произведение. Косое произведение. Использование косого произведения для нахождения ориентированной площади | 1 | 1 | |
| | Проблемы, возникающие при работе с числами с плавающей точкой, сравнение подобных чисел. Взаимное расположении точки и многоугольника: метод площадей, метод трассировки лучей, метод ближайшей точки. Формула Пика нахождения площадки решётчатого многоугольника | 1 | 1 | |
| | Выпуклые многоугольники. Построение выпуклой оболочки для набора точек | 1 | 1 | |
| **II** | **Длинная арифметика** | **1** | **3** | **4** |
| | Понятие длинного числа. Хранение и изменение длинных чисел. Умножение длинного числа на короткое. Сложение, сравнение и разность длинных чисел | 1 | 1 | |

| | | | | |
|---|---|---|---|---|
| | Произведение длинных чисел. Деление длинного числа на короткое с остатком. Реализация своего интерфейса длинного числа | | 2 | |
| **III** | **Расширение отдельных изученных олимпиадных идей** | **6** | **6** | **12** |
| | Конструктив и задачи на эффективную реализацию. Жадные алгоритмы и сортировки | | 2 | |
| | Линейные алгоритмы. Использование частичных сумм. Задача поиска подотрезка массива с максимальной суммой. Задача нахождения K-й порядковой статистики | 1 | 1 | |
| | Введение в дискретную математику. Основы булевой алгебры. Побитовые операции в программировании. Возможности их применения | 2 | | |
| | Перебор. Применение побитовых операций для полного перебора подмножеств. Битовые маски | 1 | 1 | |
| | Два указателя и двоичный поиск. Двоичный поиск по ответу. Разбор более трудных задач, решаемых двоичным поиском по ответу | 1 | 1 | |
| | Алгоритм тернарного поиска экстремума унимодальной функции | 1 | 1 | |
| **IV** | **Динамическое программирование** | **6** | **14** | **20** |
| | Теоретическая основа динамического программирования. Оптимальная структура. Принцип оптимальности на префиксе | 1 | 1 | |
| | Простейшие задачи двумерного ДП. Задачи на подсчёт количества путей в табличках и лабиринтах | | 2 | |
| | Простейшие задачи двумерного ДП. Задачи на подсчёт количества или существования комбинаций с определёнными ограничениями на их построение. Выбор аргументов для динамического программирования | | 2 | |
| | Принцип оптимальности на подотрезках. Задача расстановки знаков в выражении. Задача нахождения расстояния Левенштейна | 1 | 1 | |
| | ДП на подотрезках. Задача наибольшей общей подпоследовательности. Задача наибольшей подпоследовательности-палиндрома | | 2 | |

| | | | | |
|---|---|---|---|---|
| | Использование двоичного поиска в динамическом программировании для улучшения асимптотики. Оптимизация задачи наибольшей возрастающей подпоследовательности. Задача наибольшей общей возрастающей подпоследовательности | 1 | 1 | |
| | Использование топологической сортировки. Динамическое программирование на графах. Кратчайший путь в ациклическом графе. Количество путей в ациклическом графе | 1 | 1 | |
| | Простейшие задачи ДП на подмножествах | | 2 | |
| | Задача о 0-1 рюкзаке. Задача о неограниченном рюкзаке | 1 | 1 | |
| | Динамическое программирование по профилю. Задачи замощения доски доминошками | 1 | 1 | |
| **V** | **Алгоритмы на графах. Кратчайшие пути** | **3** | **7** | **10** |
| | Алгоритм Флойда-Уоршелла нахождения всевозможных кратчайших путей. Его использование на невзвешенном графе. Идея веса ребра, алгоритм Флойда-Уоршелла на взвешенном графе | 1 | 1 | |
| | Обход в ширину на 0-1 графе. Обход в ширину на 1..k-графе с натуральным весом рёбер | | 2 | |
| | Алгоритм Дейкстры нахождения кратчайшего взвешенного пути. Алгоритм Дейкстры с логарифмической асимптотической сложностью | 1 | 1 | |
| | Хранение графа списком рёбер. Алгоритм Беллмана-Форда нахождения кратчайшего взвешенного пути. Нахождение цикла отрицательного веса | 1 | 1 | |
| | Решение различных задач на кратчайшие пути на взвешенных графах | | 2 | |
| **VI** | **Алгоритмы на графах. Поиск в глубину и его применения** | **2** | **8** | **10** |
| | Отношение сильной связности. Разбиение ориентированного графа на компоненты сильной связности | 1 | 1 | |
| | Задачи на применение разбиения на компоненты сильной связности. Построение конденсации графа и её применение | | 2 | |

| | | | | |
|---|---|---|---|---|
| | Вершинная и рёберная связность. Важные вершины и рёбра: точки сочленения и мосты. Изучение свойств дерева обхода в глубину, лемма о белых путях | 1 | 1 | |
| | Применение алгоритма поиска мостов | | 2 | |
| | Применение алгоритма поиска точек сочленения | | 2 | |
| **VII** | **Алгоритмы на строках** | **6** | **10** | **16** |
| | Строки из ЯП Си. Указатели. Работа с си-строками, функции библиотеки <cstring> | 1 | 1 | |
| | Разбор строк и лексический анализ. Перечисления в C++, шаблон «enum» | 1 | 1 | |
| | Синтаксический анализ. Метод рекурсивного спуска | 1 | 1 | |
| | Задача поиска подстроки в строке. Наивный алгоритм и возможности его улучшения. Префиксы и суффиксы | | 2 | |
| | Z-функция и её эффективное вычисление | 1 | 1 | |
| | Алгоритм Кнута-Мориса-Пратта | | 2 | |
| | Префикс-функция и её эффективное вычисление. Построение Z-функции по префикс-функции и наоборот | 1 | 1 | |
| | Решение задачи количества уникальных подстрок в строке с помощью префикс-функции | 1 | 1 | |
| **VIII** | **Алгебра, теория чисел и дискретная математика** | **10** | **10** | **20** |
| | Матрицы в линейной алгебре. Сложение, умножение матриц, возведение квадратной матрицы в натуральную степень. Определитель квадратной матрицы | 2 | | |
| | Бинарное возведение числа в натуральную степень. Бинарное возведение квадратной матрицы в натуральную степень. Бинарный алгоритм Евклида | | 2 | |
| | Применение матриц в программировании. Количество путей фиксированной длины в графе. Быстрое вычисление K-го числа Фибоначчи | | 2 | |
| | Расширенный алгоритм Евклида. Линейные диофантовы уравнения с двумя переменными | 1 | 1 | |
| | Кольцо вычетов по модулю. Нахождение обратного элемента в кольце вычетов по модулю | 1 | 1 | |

| | | | | |
|---|---|---|---|---|
| | Китайская теорема об остатках. Алгоритм Гарнера. Функция Эйлера и её вычисление | 2 | | |
| | Коды Грея. Коды Грея для перестановок. Задача о Ханойских башнях | 1 | 1 | |
| | Полиномиальный хеш, хеш-функция. Предподсчёт в алгоритме хеширования. Вычисление хеша подстроки | 1 | 1 | |
| | Коллизии и их минимизация. Хеш-таблица как способ реализации ассоциативного массива. Методы разрешения коллизий в хеш-таблице | 2 | | |
| | Алгоритм Рабина-Карпа поиска подстроки в строке. Решение задачи количества уникальных подстрок в строке с помощью хеширования. Функция «std::unique» | | 2 | |
| **IX** | **Структуры данных** | **14** | **16** | **30** |
| | Двоичное дерево поиска. Сбалансированные деревья, двоичные кучи. Биномиальная куча, фибоначчиева куча. Приоритетные очереди | 2 | | |
| | Реализация двоичной кучи и её использование в задачах | | 2 | |
| | Декартово дерево. Декартово дерево по неявному ключу | 1 | 1 | |
| | Идея "+1/–1" массовой модификации подотрезка за O(1) | 1 | 1 | |
| | Очередь с хранением локального максимума / минимума для его нахождения за O(1) | 1 | 1 | |
| | Задачи RMQ, RSQ. Корневая эвристика | 1 | 1 | |
| | Корневая эвристика с операциями удаления и вставки в массив | 1 | 1 | |
| | Алгоритм Мо | 1 | 1 | |
| | Система непересекающихся множеств. Наивная реализация, эвристики для её улучшения | 2 | | |
| | Остовное дерево. Минимальное остовное дерево, алгоритмы его нахождения. Алгоритм Краскала. Алгоритм Прима | 1 | 1 | |
| | Решение задач на остовные деревья и СНМ | | 2 | |
| | Решение задачи RMQ с помощью разреженной таблицы | 1 | 1 | |
| | Дерево отрезков: построение и реализация запросов | 1 | 1 | |

| | | | | |
|---|---|---|---|---|
| | Решение задач RMQ, RSQ, RGQ и других с помощью построения деревьев отрезков | | 2 | |
| | Операция массового обновления на дереве отрезков. Ленивое распространение | 1 | 1 | |
| **X** | **Графы-деревья** | **6** | **8** | **14** |
| | Особенности алгоритмов обхода для деревьев. "Подвешивание" дерева, корневые деревья. Распространение на корневом дереве | 1 | 1 | |
| | Нахождение диаметра дерева. Центральные вершины дерева. Их использование в решении задач | | 2 | |
| | Модификации алгоритма обхода в глубину на дереве. Подсчёт размеров поддеревьев, порядка входа и выхода из вершин. Задача о сумме чисел в поддереве с модификацией. Проверка на предка за O(1) | 1 | 1 | |
| | Ближайший общий предок. Эйлеров обход дерева и сведение задачи LCA к задаче RMQ и дереву отрезков | 1 | 1 | |
| | Метод двоичного подъёма с динамическим препроцессингом. Нахождение ближайшего общего предка методом двоичного подъёма | 1 | 1 | |
| | Нахождение центроида дерева. Центроидная декомпозиция | 1 | 1 | |
| | Динамическое программирование на поддеревьях. Использование LCA в ДП на поддеревьях. Расширение идеи "+1/–1" для массовой модификации простых цепей на дереве | 1 | 1 | |