

Министерство образования Края
Филиал АНО ДТ «Красноярский «Кванториум» в г. Норильске
«Центр цифрового образования детей IT-Куб г. Норильск»

СОГЛАСОВАНО:

Председатель методического совета
_____ Д.С. Симашова
протокол № 14
от «07» 09 2026 г.

УТВЕРЖДАЮ:

Директор филиала
_____ Е. С. Тадер
Приказ № 02-02-95
от «07» 09 2026 г.



**ДОПОЛНИТЕЛЬНАЯ ОБЩЕОБРАЗОВАТЕЛЬНАЯ
ОБЩЕРАЗВИВАЮЩАЯ ПРОГРАММА НА 2026– 2027 уч. год**
технической направленности

«Программирование C++»

Форма реализации программы – очная

Срок реализации – 1 год

Возраст обучающихся – 14-17 лет

Составитель программы:

Шуалиев Тамерлан Арсланович

педагог дополнительного образования

г. Норильск, 2026 г.

1. ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Дополнительная общеобразовательная программа «Основы языка C++» (далее - программа) имеет техническую направленность, и разработана в соответствии с основными нормативно-правовыми документами: Федеральным Законом «Об образовании» от 29.12.2012 г. № 273-ФЗ; Порядком организации и осуществления образовательной деятельности по дополнительным общеобразовательным программам от 09.11.2018 г. № 196; Целевой моделью развития региональных систем дополнительного образования детей от 03.09.2019 г. № 467; Концепцией развития дополнительного образования детей до 2030 года утвержденной распоряжением Правительства Российской Федерации от 31.03.2022 №678-р; Санитарно-эпидемиологическими требованиями к организациям воспитания и обучения, отдыха и оздоровления детей и молодежи от 28.09.2020 г. № 28.

Уровень программы — базовый, что означает углубление интересов, расширение спектра специальных знаний в области программирования, обеспечивает трансляцию общей и целостной картины в рамках содержательно-тематического направления программы.

Назначение программы — обучить детей (школьников) интеллектуальной и творческой деятельности, а также подготовить для них фундамент для успешной сдачи Единого Государственного экзамена (ЕГЭ). В рамках курса обучающиеся овладеют продвинутыми навыками работы с языком C++, научатся эффективно применять различные структуры данных и алгоритмы, а также разовьют креативное мышление при решении сложных задач.

1.1. НОВИЗНА ДООП

Программа курса обеспечивает знакомство учащихся с фундаментальными понятиями алгоритмизации и программирования на доступном уровне, сочетая теоретические основы с практической направленностью, ориентированной на реальные, возрастно-соответствующие потребности обучающихся; она охватывает как развитие алгоритмического мышления, так и применение полученных знаний для решения задач из различных предметных областей, учитывает имеющийся парк вычислительной техники и существующие ограничения, допускает гибкую адаптацию в зависимости от уровня подготовки и интеллектуальных возможностей учащихся — как в групповом, так и в индивидуальном формате, включая возможность индивидуальной работы с каждым учеником; практическая значимость курса заключается в том, что он способствует более успешному освоению программирования за счёт развития самостоятельности обучающихся и оптимизации используемых методов и средств обучения, а отдельные элементы программы могут быть рекомендованы учителям информатики для применения на лабораторно-практических и практических занятиях.

1.2. АКТУАЛЬНОСТЬ ДООП

В условиях стремительного развития цифровых технологий и

обновления образовательных стандартов всё меньше учебного времени в школе отводится на освоение фундаментальных основ программирования. Современные пользовательские интерфейсы, построенные на принципах максимальной простоты и визуальной наглядности, формируют у школьников привычку к пассивному потреблению цифровых сервисов без понимания того, как они устроены. Удобные инструменты для поиска информации и выполнения повседневных задач в интернете снижают потребность в развитии логического мышления, аналитических способностей и навыков решения сложных задач – тех компетенций, которые лежат в основе программирования.

В результате в большинстве школ отсутствует систематическая работа по формированию у учащихся 7–11 классов прочных навыков программирования. Между тем именно глубокое понимание принципов работы программного кода, умение проектировать алгоритмы и эффективно управлять ресурсами компьютера остаются ключевыми компетенциями в современном цифровом мире.

В этом контексте особую значимость приобретают дополнительные общеобразовательные общеразвивающие программы, направленные на изучение языка C++. Этот язык, сочетающий низкоуровневый контроль над системными ресурсами с возможностями высокоуровневого программирования, позволяет учащимся не просто писать код, а понимать, как устроены программы «изнутри». Освоение C++ способствует формированию прочной базы для дальнейшего изучения других языков и технологий, развивает дисциплину мышления, внимание к деталям и культуру написания качественного программного кода.

Программа дополняет школьный курс информатики, компенсируя недостаток времени на практическую работу с реальным кодом, и создаёт условия для осознанного, системного и последовательного освоения программирования. В условиях растущего спроса на IT-специалистов владение таким языком, как C++, становится не только важным образовательным результатом, но и значимым шагом в профессиональном самоопределении учащихся.

1.3. ПЕДАГОГИЧЕСКАЯ ЦЕЛЕСООБРАЗНОСТЬ

Программа направлена на систематическое освоение языка программирования C++ и развитие у учащихся устойчивых навыков написания эффективного, структурированного и понятного кода. Обучение строится вокруг практических задач, охватывающих ключевые темы языка — от базового синтаксиса до работы с памятью, указателями и динамическими структурами данных. Важной формой работы являются регулярные практические тренировки и тематические задания, позволяющие обучающимся закреплять знания, применять их в новых условиях и объективно отслеживать свой прогресс.

Курс рассчитан на школьников, заинтересованных в глубоком изучении программирования и стремящихся не просто освоить язык, но и научиться решать сложные алгоритмические задачи. Задача педагога — сформировать у учащихся навыки самостоятельной теоретической и практической работы, научить их анализировать условия задач, проектировать решения и доводить код до рабочего состояния.

При разработке программы учтены возрастные особенности обучающихся: рост познавательной самостоятельности, устойчивость интересов, стремление к профессиональному самоопределению и активный интерес к самообразованию.

Реализация программы способствует решению ключевых педагогических задач: развитию алгоритмического и логического мышления, формированию прочных компетенций в программировании на C++, а также повышению внутренней мотивации к освоению IT-дисциплин. Структурированный подход, основанный на последовательном усложнении задач и постоянной практике, развивает умение самостоятельно анализировать проблемы, находить недостающую информацию и разрабатывать эффективные, оптимизированные решения.

1.4. ЦЕЛЬ ДООП

Цель программы – формирование у обучающихся прочных и актуальных знаний о языке программирования C++, понимания его возможностей, а также развитие навыков практического применения этих знаний для решения олимпиадных и прикладных задач, включая подготовку и публичную презентацию собственных проектов.

Задачи программы:

- –познакомить обучающихся с основами программирования и логикой построения алгоритмов;
- –изучить базовый и продвинутый синтаксис языка C++;
- –сформировать устойчивое алгоритмическое мышление;
- –освоить ключевые алгоритмы и методы, применяемые при решении олимпиадных задач по информатике;
- –изучить динамические структуры данных (списки, стеки, очереди, деревья и др.) и научиться эффективно работать с ними на C++;
- –развить навыки управления памятью, включая понимание работы с указателями, динамическим выделением памяти и оптимизацией ресурсов;
- –познакомить с элементами олимпиадной математики, необходимыми для анализа и построения эффективных решений;
- –сформировать основы информационной культуры, включая ответственное и этичное обращение с информацией;
- –воспитать потребность в самостоятельном поиске, освоении и применении знаний из дополнительных источников;
- –развить способность формулировать гипотезы, проектировать стратегии решения задач, прогнозировать и анализировать результаты, а также

- оптимизировать и детализировать собственные алгоритмы;
- –сформировать устойчивый интерес к IT-сфере в целом, поддержать позитивное отношение к различным направлениям индустрии и мотивировать на дальнейшее профессиональное развитие, независимо от уровня достижений в олимпиадном программировании.

1.5. ВОЗРАСТ ОБУЧАЮЩИХСЯ, УЧАСТВУЮЩИХ В ДООП

Программа «Программирование на языке C++» рассчитана на обучающихся 14-17 лет. Максимальное количество обучающихся в группе – 12 человек.

1.6. УСЛОВИЯ ВХОЖДЕНИЯ В ДООП

Набор на программу осуществляется путём договорённости и переговорных мероприятий с общеобразовательными учреждениями, в том числе со школами, гимназиями, лицеями.

1.7. СРОК РЕАЛИЗАЦИИ ДООП

Программа рассчитана на 1 учебный год. Нагрузка на обучающегося составляет 144 часа за весь период обучения.

1.8. РЕЖИМ ЗАНЯТИЙ, ФОРМЫ И МЕТОДЫ ОБУЧЕНИЯ

Учебные занятия проходят в очной форме. Режим занятий – 2 раза в неделю по 2 академических часа (1 академический час - 40 минут) с обязательным перерывом.

При проведении занятий используются как стандартные формы работы, так и нетрадиционные:

- лекции: изложение преподавателем информации, необходимой для введения в новые темы и предоставления основных концепций;
- семинары: краткий повтор теоретической части лекции и ее практические групповые обсуждения, совместное решение проблемных задач;
- практические занятия: работа с конкретными задачами, необходимая для выработки практических умений;
- дискуссии: постановка спорных вопросов, отработка умения отстаивать и аргументировать свою точку зрения;
- круглый стол: неформальное обсуждение выбранной тематики;
- мозговая атака: решение нестандартных задач в коллективе.

Обучение проводится в комбинированном формате с проведением лекций и демонстраций для наилучшего усвоения теоретического материала. После лекционных занятий проводятся практические, направленные на закрепление полученных теоретических знаний.

1.9. ОЖИДАЕМЫЕ ОБРАЗОВАТЕЛЬНЫЕ РЕЗУЛЬТАТЫ И

ЭФФЕКТЫ, СПОСОБЫ ПРЕДЪЯВЛЕНИЯ И ОТСЛЕЖИВАНИЯ РЕЗУЛЬТАТОВ

Предметные результаты:

- – умение понимать, объяснять и применять на практике конструкции языка C++ различного уровня сложности;
- – уверенное владение синтаксисом и семантикой C++;
- – развитое алгоритмическое мышление, позволяющее эффективно анализировать и проектировать решения задач;
- – навык написания читаемого, корректного, оптимизированного и структурированного кода на C++;
- – свободное использование стандартной библиотеки C++ (STL), включая контейнеры, алгоритмы и итераторы;
- – значительный практический опыт программирования на C++ в условиях, приближённых к олимпиадным;
- – сформированный навык решения олимпиадных задач по информатике среднего и повышенного уровня сложности;
- – умение применять динамические структуры данных — списки, стеки, очереди, деревья, графы — для решения разнообразных алгоритмических задач;
- – понимание и применение методов динамического программирования;
- – освоение базовых алгоритмов олимпиадного программирования: сортировки, поиска, обхода графов, работы со строками и др.

Метапредметные результаты:

- сформированное ответственное отношение к информации, учитывающее правовые и этические нормы её использования и распространения.

Личностные результаты:

- развитая потребность в самостоятельном поиске, освоении и применении знаний из дополнительных источников;
- способность формулировать гипотезы, разрабатывать стратегию решения задачи, прогнозировать и анализировать результаты, а также находить рациональные и оптимальные пути решения через детализацию и улучшение алгоритмов.

Система оценки результатов:

Текущий контроль осуществляется в ходе занятий посредством наблюдения, устных и письменных опросов, а также выполнения практических и тренировочных заданий.

Промежуточная аттестация проводится в конце первого полугодия в форме комплексной проверочной работы, включающей как теоретические вопросы, так и практическое задание (продолжительность — 2 часа).

Итоговая аттестация проходит в виде защиты собственного программного проекта, разработанного обучающимся в ходе освоения программы. Оценка основывается на корректности реализации, использовании изученных конструкций языка C++, структуре кода, а также

способности учащегося объяснить логику и архитектуру своего решения.

Дополнительно результаты освоения программы могут подтверждаться участием в научно-технических мероприятиях, конкурсах, выставках проектов, а также публичной презентацией разработанных программных продуктов.

2. УЧЕБНО-ТЕМАТИЧЕСКИЙ ПЛАН

№	Наименование разделов	Всего часов	Теория	Практика	Форма контроля
1	Основы программирования и языка C++	40	16	24	
1.1	Введение в программирование: алгоритм, программа, компиляция. Среда разработки	2	2	0	
1.2	Ввод и вывод данных: потоки cin, cout, форматирование	4	2	2	
1.3	Основы C++: синтаксис, переменные, операторы, условные конструкции, циклы	10	4	6	
1.4	Функции: объявление, параметры, возврат значений	10	2	8	
1.5	Массивы и векторы: одномерные и многомерные, std::vector, итерирование	10	4	6	
1.6	Текущий контроль по разделу «Основы программирования и языка C++»	4	2	2	Защита мини-проекта + тест по основам языка
2	Углублённое программирование на C++	56	20	36	
2.1	Указатели и ссылки: адреса, разыменование, передача аргументов	10	4	6	
2.2	Динамическое выделение памяти: new/delete, управление ресурсами	10	4	6	
2.3	Структуры и основы ООП: объявление, поля, методы, инкапсуляция	12	4	8	
2.4	Стандартная библиотека C++ (STL): контейнеры (vector, set, map), алгоритмы, итераторы	10	4	6	
2.5	Отладка и тестирование: исключения, отладчик, логирование, проверка корректности	10	4	6	
2.6	Текущий контроль по разделу «Углублённое	4	2	2	Защита своего проекта +

	программирование на С++»				письменный тест
3	Практическое применение С++	44	22	22	
3.1	Работа с файлами: чтение/запись текстовых и бинарных данных	10	6	4	
3.2	Рекурсия: принцип, примеры (факториал, Фибоначчи, обход)	10	4	6	
3.3	Основы структур данных: стек, очередь, связный список (реализация)	10	6	4	
3.4	Алгоритмическое мышление: перебор, жадные стратегии, оптимизация	10	4	6	
3.5	Текущий контроль по разделу «Практическое применение С++»	4	2	2	Публичная защита собственного проекта на С++
4	Итоговый проект и защита	4	0	4	
Итого		144	58	86	

3. СОДЕРЖАНИЕ ПРОГРАММЫ

Раздел 1. Основы программирования и языка C++ (40 часов)

Тема 1.1. Введение в программирование на C++ (2 часа)

Теория: правила техники безопасности при работе за компьютером; общее представление о программировании; структура программы на языке C++; основные типы данных (int, double, char, bool); базовые операции с данными; особенности целочисленной арифметики.

Практика: знакомство со средой разработки (IDE), создание и запуск первой программы «Hello, World!».

Тема 1.2. Ввод и вывод данных: работа с потоками (4 часа)

Теория: языковые конструкции C++: cin, cout; операторы << и >>; понятие потоков ввода-вывода; чтение и запись данных из/в файлы; метод get(); введение в строковые потоки (stringstream).

Практика: реализация программ с пользовательским вводом и выводом; обработка входных и выходных данных; составление линейных алгоритмов; работа с текстовыми файлами.

Тема 1.3. Основы C++: синтаксис, переменные, операторы, условия, циклы (10 часов)

Теория: синтаксис языка C++; объявление и инициализация переменных; арифметические, логические и сравнительные операторы; условные конструкции (if, else if, else, switch); циклы (for, while, do-while); алгоритмы: проверка числа на простоту, разложение на простые множители, НОД и НОК; понятие временной сложности алгоритма; однопроходные алгоритмы.

Практика: реализация эффективных алгоритмов проверки простоты и разложения чисел; применение НОД/НОК в задачах; оптимизация циклических конструкций – замена вложенных циклов формулами или уменьшение глубины вложенности.

Тема 1.4. Функции: объявление, параметры, возврат значений (10 часов)

Теория: объявление и вызов функций; передача параметров по значению и по ссылке; возвращаемые значения; пользовательские функции; рекурсия как метод решения задач; работа со строками через функции.

Практика: структурирование программ с использованием функций; решение задач путём декомпозиции на подзадачи; написание рекурсивных функций (например, вычисление факториала, чисел Фибоначчи).

Тема 1.5. Массивы и векторы: объявление, индексация, итерирование, многомерные массивы (10 часов)

Теория: статические массивы и динамические контейнеры (std::vector); индексация и перебор элементов; многомерные массивы; префиксные суммы;

сортировка подсчётом; стандартная сортировка из библиотеки <algorithm> (включая использование пользовательских компараторов); поиск минимума и максимума; перестановка элементов; вывод массива; инициализация массивов константами.

Практика: сортировка и обработка массивов; применение префиксных сумм для ускорения вычислений; решение задач с использованием сортировки подсчётом и стандартных алгоритмов STL.

Текущий контроль по разделу «Основы программирования и языка C++» (4 часа)

Теория и практика: теоретическое тестирование по пройденным темам; выполнение и защита мини-проекта (например, калькулятор, обработчик списков, статистический анализ данных).

Раздел 2. Углублённое программирование на C++ (56 часов)

Тема 2.1. Указатели и ссылки (10 часов)

Теория: понятие адреса памяти; операторы & и *; указатели и ссылки; передача аргументов в функции по указателю и по ссылке; различия между указателями и ссылками.

Практика: написание функций, изменяющих данные через указатели/ссылки; отладка программ с использованием адресов памяти.

Тема 2.2. Динамическое управление памятью (10 часов)

Теория: операторы new и delete; динамическое выделение памяти для переменных и массивов; утечки памяти; правила RAII (Resource Acquisition Is Initialization); базовые принципы безопасной работы с памятью.

Практика: создание и освобождение динамических структур; написание программ с контролем использования памяти.

Тема 2.3. Структуры и основы объектно-ориентированного программирования (12 часов)

Теория: объявление и использование структур (struct); поля и методы; инкапсуляция; введение в классы; конструкторы; принципы проектирования простых объектов.

Практика: моделирование реальных сущностей («Книга», «Автомобиль», «Студент»); создание программ с использованием пользовательских типов данных.

Тема 2.4. Стандартная библиотека шаблонов (STL) (10 часов)

Теория: контейнеры: vector, set, map, queue, stack; итераторы; алгоритмы: sort, find, count, accumulate; работа с парами (pair); функциональные объекты и лямбда-выражения (вводно).

Практика: решение задач с использованием STL; обработка коллекций данных; эффективная работа с ассоциативными контейнерами.

Тема 2.5. Отладка, тестирование и обработка ошибок (10 часов)

Теория: типы ошибок (синтаксические, логические, времени выполнения); отладка с помощью отладчика; логирование; исключения (try, catch, throw); написание тестируемых функций.

Практика: поиск и исправление ошибок в готовом коде; написание модульных тестов; обработка исключительных ситуаций.

Текущий контроль по разделу «Углублённое программирование на C++» (4 часа)

Теория и практика: письменный тест по ключевым понятиям; защита среднего проекта (например, система учёта, простой симулятор, обработчик данных).

Раздел 3. Практическое применение C++ (44 часов)

Тема 3.1. Работа с файлами (10 часов)

Теория: потоки файлового ввода-вывода (ifstream, ofstream, fstream); текстовые и бинарные файлы; сериализация простых структур данных.

Практика: создание программ для сохранения и загрузки данных (например, адресная книга, дневник, журнал событий).

Тема 3.2. Рекурсия (8 часов)

Теория: принцип рекурсии; базовый случай и рекурсивный вызов; стек вызовов; примеры: факториал, числа Фибоначчи, обход деревьев, генерация перестановок.

Практика: реализация рекурсивных алгоритмов; сравнение с итеративными аналогами.

Тема 3.3. Основы структур данных (10 часов)

Теория: стек, очередь, связный список – принципы работы и реализация на C++ (через массивы и указатели); сравнение с STL-аналогами.

Практика: самостоятельная реализация структур данных; применение в задачах (например, проверка скобок, очередь задач).

Тема 3.4. Алгоритмическое мышление и оптимизация решений (12 часов)

Теория: полный перебор, жадные алгоритмы, идея оптимизации; оценка эффективности решений; выбор подходящего метода в зависимости от условий задачи.

Практика: решение прикладных задач (планирование, маршрутизация, распределение ресурсов); рефакторинг кода для повышения производительности.

Текущий контроль по разделу «Практическое применение C++» (4 часа)

Теория и практика: презентация итогового проекта первой версии; обсуждение архитектуры и плана доработки.

Раздел 4. Итоговый проект и защита (4 часа)

Практика: финальная доработка, тестирование и публичная защита собственного программного проекта на C++. Оценка включает корректность работы, качество кода, использование изученных возможностей языка и умение объяснить своё решение.

4. МАТЕРИАЛЬНО-ТЕХНИЧЕСКАЯ И ИНФОРМАЦИОННО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ

Материально-техническое обеспечение:

1. Стол и стул преподавателя
2. Столы и стулья для обучающихся (по количеству участников группы)
3. Рабочая станция преподавателя (стационарный компьютер или ноутбук)
4. Ноутбуки или компьютеры для обучающихся (1 устройство на учащегося)
5. Интерактивная доска или проектор с экраном
6. Многофункциональное устройство (МФУ) для печати и сканирования
7. Достаточное количество розеток и точек подключения к электрической сети
8. Стабильное подключение к локальной сети и интернету (для доступа к справочным материалам, онлайн-компиляторам и репозиториям)

Программное обеспечение:

1. Операционная система – Windows 10/11 или Linux (допускается использование macOS при наличии)
2. Интегрированная среда разработки (IDE) для C++:
 - Visual Studio Community (с поддержкой C++)
 - или Code::Blocks
 - или CLion (при наличии лицензии)
 - альтернативно – онлайн-платформы: Replit, Compiler Explorer, OnlineGDB
3. Компилятор GCC (в составе MinGW или как часть Linux-системы)
4. Пакет офисных приложений (MS Office или LibreOffice) – для оформления отчётов и документации к проектам
5. Доступ к справочным ресурсам по языку C++ (например, cppreference.com, официальная документация)

5. РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

Основная литература:

- 1) Томас Х. Кормен, Алгоритмы. Вводный курс. – Вильямс. 2014. – 208 с.
- 2) Златопольский Д.М. Программирование: типовые задачи, алгоритмы, методы. – М.: Бином. Лаборатория знаний, 2007. – 223 с.
- 3) Великович Л.С., Цветкова М.С. Программирование для начинающих. – М. Бином. Лаборатория знаний, 2007. – 287 с.
- 4) Стефан Рэнди Дэвис C++ For Dummies, . – Диалектика. 2018. – 336 с.
- 5) Липпман С.Б., Лажоие Ж, Му Б.Э. Язык программирования C++. Базовый курс. – Вильямс. 2014. – 1120 с.

6. СВЕДЕНИЯ О СОСТАВИТЕЛЯХ ДООП

Шуалиев Тамерлан Арсланович, педагог дополнительного образования, преподаватель по направлению «Программирование на языке C++». Образование незаконченное высшее.

7. МАТЕРИАЛЫ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ ПО РАЗДЕЛАМ ПРОГРАММЫ И ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ

Перечень допустимых тем мини-проектов, проектов и проектных задач

1. «Интерактивный калькулятор» — программа с текстовым меню, поддерживающая арифметические операции, ввод/вывод и базовую обработку ошибок (деление на ноль и др.) (раздел 1).
2. «Адресная книга» — приложение для добавления, поиска, редактирования и удаления контактов с сохранением данных в текстовый файл (раздел 3).
3. «Статистический анализ массива» — программа для вычисления среднего, медианы, минимума, максимума, частоты элементов и построения простой текстовой гистограммы (разделы 1–2).
4. «Симулятор очереди задач» — реализация очереди (FIFO) с возможностью добавления, извлечения и просмотра текущего состояния (раздел 3).
5. «Викторина» — интерактивная программа с вопросами, подсчётом баллов и выводом результата; вопросы могут загружаться из файла (разделы 1, 3).
6. «Конвертер единиц» — приложение для перевода между единицами измерения (длина, масса, температура и др.) с выбором через меню (раздел 1).
7. «Генератор паролей» — программа, создающая случайные пароли заданной длины с выбором набора символов (цифры, буквы, спецсимволы) (раздел 1).
8. «Дневник расходов» — система учёта личных трат с возможностью добавления записи, просмотра за день/неделю и сохранения в файл (раздел 3).
9. «Текстовый редактор (упрощённый)» — программа для чтения, редактирования и сохранения небольших текстовых файлов (раздел 3).
10. «Игра «Угадай число»» — классическая игра с генерацией случайного числа, подсказками и подсчётом попыток (раздел 1).
11. «Система голосования» — приложение для проведения опроса с выбором из нескольких вариантов и выводом итоговой статистики (разделы 1–2).
12. «Календарь событий» — программа для добавления, просмотра и удаления событий по датам с сохранением в файл (раздел 3).
13. «Библиотечный каталог» — учёт книг с возможностью поиска по

- автору, названию, году издания (*разделы 2–3*).
14. «Симулятор стека» — реализация структуры данных «стек» (LIFO) с операциями push/pop и визуализацией состояния (*раздел 3*).
 15. «Анализатор текста» — программа, подсчитывающая количество слов, строк, символов, частоту букв или слов в тексте (*разделы 1–2*).
 16. «Шифратор/дешифратор» — реализация простого шифра (например, Цезаря) с вводом текста и ключа (*раздел 1*).
 17. «Справочник по языкам программирования» — интерактивный каталог с краткой информацией о языках, возможностью поиска и фильтрации (*разделы 2–3*).
 18. «Трекер привычек» — приложение для отслеживания ежедневных привычек (например, «пить воду», «читать») с отметками и статистикой (*раздел 3*).
 19. «Мини-игра «Поле чудес»» — упрощённая версия игры с загаданным словом, попытками угадывания букв и подсчётом очков (*разделы 1–2*).
 20. Собственный итоговый проект — индивидуальная разработка по выбору обучающегося (с одобрения педагога), демонстрирующая комплексное применение знаний: работа с функциями, массивами/векторами, файлами, структурами данных и основами проектирования программ (*все разделы*).

Тест №1. Основы программирования и языка C++

(20 вопросов)

1. Какой тип данных в C++ используется для хранения целого числа?

- a) double
- b) bool
- c) int
- d) char

2. Что выведет следующий фрагмент кода?

```
int x = 5;  
if (x > 3) cout << "Да";  
else cout << "Нет";
```

- a) Нет
- b) Да
- c) Ошибка компиляции
- d) Ничего не выведет

3. Какой оператор используется для ввода данных с клавиатуры?

- a) cin
- b) cout
- c) scanf
- d) input

4. Как объявить массив из 10 целых чисел?

- a) `int arr[10];`
- b) `array<int, 10> arr;`
- c) `vector<int> arr(10);`
- d) Все вышеперечисленные варианты корректны

5. Что делает цикл `for (int i = 0; i < 5; i++)`?

- a) Выполняется 5 раз
- b) Выполняется 6 раз
- c) Выполняется бесконечно
- d) Не выполняется ни разу

6. Какой заголовочный файл необходим для работы с `vector`?

- a) `<iostream>`
- b) `<vector>`
- c) `<array>`
- d) `<math.h>`

7. Выберите верные утверждения:

- a) Функция может не возвращать значение
- b) Функция `main` обязательна в программе
- c) Функция может вызывать саму себя
- d) Все функции должны быть объявлены до `main`

8. Что выведет программа?

```
int a = 10, b = 3;  
cout << a % b;
```

- a) 3
- b) 1
- c) 0
- d) 10

9. Какой оператор используется для логического «И»?

- a) `&`
- b) `&&`
- c) `and`
- d) `||`

10. Что такое `std::vector`?

- a) Статический массив
- b) Динамический массив из стандартной библиотеки
- c) Указатель на массив
- d) Функция для сортировки

11. Как передать значение в функцию по ссылке?

- a) `void f(int x)`
- b) `void f(int& x)`
- c) `void f(int* x)`
- d) Невозможно

12. Какой результат выполнения?

```
int x = 7;  
cout << (x % 2 == 0 ? "Чётное" : "Нечётное");
```

- a) Чётное
- b) Нечётное
- c) 7
- d) Ошибка

13. Какой цикл лучше использовать, если количество итераций неизвестно заранее?

- a) for
- b) while
- c) do-while
- d) Любой из перечисленных

14. Что выведет код?

```
vector<int> v = { 1, 2, 3 };  
cout << v.size();
```

- a) 2
- b) 3
- c) 4
- d) Ошибка

15. Какой тип данных подходит для хранения символа?

- a) string
- b) char
- c) int
- d) bool

16. Что делает оператор break в цикле?

- a) Переходит к следующей итерации
- b) Прерывает выполнение цикла
- c) Завершает программу
- d) Ничего не делает

17. Как объявить функцию, возвращающую целое число и принимающую два целых параметра?

- a) int f(int a, int b)
- b) void f(int a, int b)
- c) int f(a, b)
- d) function int(a, b)

18. Какой результат?

```
int arr[3] = { 10, 20, 30 };  
cout << arr[1];
```

- a) 10

- b) 20
- c) 30
- d) Ошибка

19. Какой заголовочный файл нужен для cout и cin?

- a) <stdio.h>
- b) <iostream>
- c) <fstream>
- d) <string>

20. Какой из перечисленных — правильный способ объявить строку?

- a) string s = "Привет";
- b) char s = "Привет";
- c) int s = "Привет";
- d) vector<char> s = "Привет";

Тест №2. Углублённое программирование на C++

(20 вопросов)

1. Что такое указатель?

- a) Переменная, хранящая значение другой переменной
- b) Переменная, хранящая адрес другой переменной
- c) Синоним переменной
- d) Функция для работы с памятью

2. Как выделить память под один int динамически?

- a) int* p = new int;
- b) int p = malloc(sizeof(int));
- c) int* p = alloc(int);
- d) int p = new int;

3. Что произойдёт, если не освободить память, выделенную через new?

- a) Программа завершится с ошибкой
- b) Возникнет утечка памяти
- c) Ничего — система сама освободит
- d) Компилятор выдаст предупреждение

4. Как передать массив в функцию наиболее безопасно в современном C++?

- a) Через указатель
- b) Через std::vector по константной ссылке
- c) Через глобальную переменную
- d) Через int arr[]

5. Что такое struct?

- a) Тип данных для объединения нескольких переменных
- b) Функция
- c) Указатель
- d) Контейнер STL

6. Какой контейнер STL позволяет хранить пары «ключ–значение»?

- a) vector

- b) set
- c) map
- d) stack

7. Выберите верные утверждения о std::vector:

- a) Автоматически изменяет размер
- b) Хранит элементы в непрерывной памяти
- c) Поддерживает индексацию
- d) Все вышеперечисленное верно

8. Что выведет код?

```
int x = 5;  
int* p = &x;  
cout << *p;
```

- a) Адрес переменной x
- b) 5
- c) Ошибка
- d) 0

9. Как освободить память, выделенную через new int[10]?

- a) delete p;
- b) delete[] p;
- c) free(p);
- d) Ничего не нужно

10. Что такое RAII?

- a) Метод отладки
- b) Принцип управления ресурсами через время жизни объектов
- c) Библиотека для работы с файлами
- d) Тип исключения

11. Какой заголовочный файл нужен для std::map?

- a) <map>
- b) <vector>
- c) <algorithm>
- d) <utility>

12. Что делает следующий код?

```
vector<int> v = {3, 1, 4};  
sort(v.begin(), v.end());
```

- a) Сортирует по убыванию
- b) Сортирует по возрастанию
- c) Удаляет дубликаты
- d) Ничего не меняет

13. Какой тип исключения используется в C++ по умолчанию?

- a) error

- b) exception
- c) throw
- d) Исключения не используются

14. Что выведет программа?

```
try {  
    throw 42;  
} catch (int e) {  
    cout << "Ошибка: " << e;  
}
```

- a) Ошибка: 42
- b) 42
- c) Ошибка компиляции
- d) Ничего

15. Какой контейнер обеспечивает FIFO (первым пришёл — первым ушёл)?

- a) stack
- b) queue
- c) vector
- d) set

16. Что такое ссылка (int& x)?

- a) Альтернативное имя для существующей переменной
- b) Указатель на переменную
- c) Копия переменной
- d) Глобальная переменная

17. Какой метод vector возвращает количество элементов?

- a) length()
- b) size()
- c) count()
- d) capacity()

18. Какой заголовочный файл нужен для работы с файлами?

- a) <iostream>
- b) <fstream>
- c) <stdio.h>
- d) <file>

19. Что такое конструктор в структуре/классе?

- a) Функция для удаления объекта
- b) Функция для инициализации объекта при создании
- c) Метод для сортировки
- d) Указатель на объект

20. Какой из перечисленных — правильный способ открыть файл для записи?

- a) ofstream file("data.txt");
- b) ifstream file("data.txt");
- c) fstream file("data.txt", ios::in);
- d) file.open("data.txt", "w");