МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«Московский физико-технический институт (национальный исследовательский университет)»

(МФТИ, Физтех)

УТВЕРЖДАЮ

Ректор МФТИ

д-р физ.-мат. наук, профессор

_____ Д. В. Ливанов

«__» _____ 2022 г.

Дополнительная общеобразовательная общеразвивающая программа «Программирование на C++»

Москва 2022

Шаблон дополнительной общеобразовательной программы

Общие данные об образовательной программе

«Программирование на C++»

Об организации

Наименование поля	Допустимые значения поля	Значение поля
ИНН организации, осуществляющей образовательную деятельность	10 арабских цифр	5008006211
Наименование организации	строка	федеральное государственное автономное образовательное учреждение высшего образования «Московский физико-технический институт (национальный исследовательский университет)»
Логотип организации	изображение в формате jpeg разрешением не менее 100x100 пиксель	
Ссылка на логотип организации	URL на изображение, находящееся в сети интернет	https://it-edu.com/_data/63209637cae37_mfti-jpeg.jpg
Контакты ответственного за программу. ФИО	строка от 5 до 255 символов	Мартемьянов Роман Юрьевич
Контакты ответственного за программу. Должность	строка от 5 до 255 символов	Заместитель директора Центра развития ИТ-образования МФТИ
Контакты ответственного за программу. Телефон	Формат +7(XXX)XXXXXXX	+7(915)0842180
Контакты ответственного за программу. E-mail	строка	martemyanov@phystech.edu

Программа

Наименование поля	Допустимые значения поля	Значение поля
Название программы (курса)	строка	Программирование на C++
Описание программы	строка не менее 1000 не более 5000 символов	Программа «Программирование на C++» имеет техническую направленность, в её основу заложены принципы модульности и практической направленности, что обеспечит вариативность обучения. Содержание учебных модулей предполагает детальное изучение алгоритмизации, реализацию межпредметных связей, организацию проектной и исследовательской деятельности обучающихся. Цель программы - формирование познавательной активности обучающихся в области функционального и объектно-ориентированного программирования, приобретение навыков работы с базовыми и сложными структурами языка в интегрированных средах разработки, получение навыков самостоятельного написания кода и разработки эффективных алгоритмов и программ. Программа включает 4 модуля: – Введение в C++ – STL и объектно-ориентированное программирование – Проектирование приложений – Оконные и специализированные приложения
Аннотация	строка до 1000 символов	Дополнительная общеобразовательная программа «Программирование на C++» от МФТИ разработана для школьников 8-11 классов в рамках проекта «Код будущего». Программа включает 4 модуля. Обучение бесплатное
Цель программы	строка не менее 100 символов	Цель программы - формирование познавательной активности обучающихся в области функционального и объектно-ориентированного программирования, приобретение навыков работы с базовыми и сложными структурами языка в интегрированных средах разработки, получение навыков самостоятельного

		написания кода и разработки эффективных алгоритмов и программ.
Актуальность	строка не менее 500 символов	В обществе всё большее значение приобретает умение человека использовать компьютер не на пользовательском уровне, а на уровне начинающего программиста. В обязательном школьном курсе информатики программирование представлено на уровне, достаточном для прохождения экзамена, но не предполагает овладение практическими навыками применения языка. Следствием этого - формальное восприятие обучающимися основ современного программирования и неумение применять полученные знания на практике. Указом Президента Российской Федерации от 07.05.2018 г. № 204 «О национальных целях и стратегических задачах развития Российской Федерации на период до 2024 года» поставлена национальная цель – обеспечение ускоренного внедрения цифровых технологий в экономике. В условиях широкого внедрения онлайн-сервисов, электронных услуг, развития цифровой экономики актуальной является проблема подготовки кадров, в том числе в области программирования на языке С++. Отечественные компании испытывают потребность в квалифицированных кадрах, способных решать прикладные задачи на языке программирования С++. Программа имеет техническую направленность, в её основу заложены принципы модульности и практической направленности, что обеспечит вариативность обучения. Содержание учебных модулей предполагает детальное изучение алгоритмизации, реализацию межпредметных связей, организацию проектной и исследовательской деятельности обучающихся.
Дополнительная информация	строка	
Формат обучения	значение из: "Онлайн"	Онлайн

	"Оффлайн" "Смашанный"	
Уровень сложности	значение из: "Начальный" "Базовый" "Продвинутый"	Начальный
Срок освоения образовательной программы	строка	17 октября 2022 года – 1 ноября 2023 года
Объем каждого модуля в ак.ч.	целое число	36
Объем часов в неделю в ак.ч	целое число	6
Минимальное количество человек на одном потоке курса	целое число < 100000	1000
Количество уроков	целое число	72
Данные о количестве школьников, ранее успешно прошедших обучение по образовательной программе	целое число, при наличии	0
Направленность программы	строка	Программирование и создание ИТ-продуктов
Язык программирования	строка	C++
Образовательная программа не представлена для участия в иных федеральных проектах, направленных на дополнительное образование граждан, кроме федерального проекта «Развитие кадрового потенциала ИТ-отрасли»	строка, значения: "Представлена"/ "Не представлена"	Не представлена
Образовательная программа не была реализована до начала отбора и/или не реализовывается в период отбора на безвозмездной основе	строка, значения: "Реализована ранее"/ "Не реализована	Не реализована
Версия программы	строка, значение: "Первая версия образовательной программы, ранее не реализованная" "Ранее реализованная версия образовательной программы"	Первая версия образовательной программы, ранее не реализованная

Категория обучающихся (возраст) по программе	строка, значение: "Учащиеся 8 класса" "Учащиеся 9 класса" "Учащиеся 10 класса" "Учащиеся 11 класса"	Учащиеся 8 класса Учащиеся 9 класса Учащиеся 10 класса
Описание планируемых результатов обучения	строка	Личностные результаты – формирование ответственного отношения к учению, готовности и способности обучающихся к саморазвитию и самообразованию на основе мотивации к обучению и познанию, осознанному выбору и построению дальнейшей индивидуальной траектории образования на базе ориентировки в мире профессий и профессиональных предпочтений, с учётом устойчивых познавательных интересов; – формирование целостного мировоззрения, соответствующего современному уровню развития науки и общественной практики, учитывающего социальное, культурное, языковое, духовное многообразие современного мира; – формирование коммуникативной компетентности в общении и сотрудничестве со сверстниками, детьми старшего и младшего возраста, взрослыми в процессе образовательной, общественно полезной, учебно-исследовательской, творческой и других видов деятельности. – развитие опыта участия в социально значимых проектах, повышение уровня самооценки благодаря реализованным проектам. Метапредметные результаты – умение самостоятельно определять цели своего обучения, ставить и формулировать для себя новые задачи в учёбе и познавательной деятельности, развивать мотивы и интересы своей познавательной деятельности; – умение самостоятельно планировать пути достижения целей, в том числе альтернативные, осознанно выбирать

		наиболее эффективные способы решения учебных и познавательных задач; – умение соотносить свои действия с планируемыми результатами, осуществлять контроль своей деятельности в процессе достижения результата, определять способы действий в рамках предложенных условий и требований, корректировать свои действия в соответствии с изменяющейся ситуацией; – умение оценивать правильность выполнения учебной задачи, собственные возможности её решения; – владение основами самоконтроля, самооценки, принятия решений и осуществления осознанного выбора в учебной и познавательной деятельности; – умение определять понятия, создавать обобщения, устанавливать аналогии, классифицировать, самостоятельно выбирать основания и критерии для классификации, устанавливать причинно-следственные связи, строить логическое рассуждение, умозаключение (индуктивное, дедуктивное и по аналогии) и делать выводы; – умение создавать, применять и преобразовывать знаки и символы, модели и схемы для решения учебных и познавательных задач; – умение организовывать учебное сотрудничество и совместную деятельность с учителем и сверстниками; работать индивидуально и в группе: находить общее решение и разрешать конфликты на основе согласования позиций и учёта интересов; формулировать, аргументировать и отстаивать своё мнение; – формирование и развитие компетентности в области использования информационно коммуникационных технологий. Предметные результаты – знание необходимой терминологии («данные», «команда», «алгоритм», «модель», «объект»,

		«техническое задание»), смысла данных понятий и умение применять полученные знания на практике; – знание об алгоритмических конструкциях и структурах данных; – знание основных понятий и этапов проектной деятельности; – умение соблюдать этикет программиста, не разрабатывать заведомо неработоспособный или приносящий вред программный код; – умение соблюдать сетевой этикет, другие базовые нормы информационной этики и права при работе с компьютерными программами и в сети Интернет; – умение составлять техническое задание на основе требований заказчика; – умение разрабатывать программные решения, осуществлять их проектирование, разработку, тестирование, отладку и внедрение; – развитие умений составить и записать алгоритм для конкретного исполнителя; – навыки пошагового выполнения алгоритмов, умение осуществлять данные операции как вручную, так и с использованием программы отладки; – навыки определения асимптотических оценок времени выполнения и затрат памяти для алгоритмов.
Ссылка на лендинг	строка	https://edu.mipt.ru/futurecode/
Ссылка на LMS	строка	https://edu.mipt.ru/member/meroprijatija/programmirovanie-na-c-dlya-shkolnikov/
Страница обучения на курсе	строка	https://edu.mipt.ru/member/meroprijatija/programmirovanie-na-c-dlya-shkolnikov/
Итоговая аттестация. Количество академических часов	Целое число больше 0	Итоговая аттестация для дополнительных общеобразовательных программ для детей и взрослых не предусмотрена Федеральным законом от 29.12.2021 г. №273-ФЗ «Об образовании в Российской Федерации» и Приказом Министерства просвещения Российской Федерации от 09.11.2018 г. № 196 «Об утверждении
Итоговая аттестация. Формы контроля	строка от 4 до 255 символов	
Итоговая аттестация. Диагностические инструменты	строка не менее 10 символов	

Итоговая аттестация. Показатели и критерии оценивания	строка не менее 50 символов	Порядка организации и осуществления образовательной деятельности по дополнительным общеобразовательным программам»
Итоговая аттестация. Шкала оценивания, нижнее значение	целое число	
Итоговая аттестация. Шкала оценивания, верхнее значение	целое число, большее чем нижнее значение шкалы	
Итоговая аттестация. Шкала оценивания, минимальный проходной балл	целое число в диапазоне шкалы	

Преподаватели

Фамилия	Имя	Отчество	Основное место работы совпадает	Наименование основного места работы	Должность	Ссылка на Веб-страницу с портфолио	Фото	Ссылка на фото	Есть ученая степень	Ученая степень	Есть ученое звание	Ученое звание	Отметка о полученном согласии на обработку персональных данных
строка от 2 до 100 символов	строка от 2 до 100 символов	строка от 2 до 100 символов	"да"/"нет"	строка от 2 до 255 символов. Обязательно, если указано, что основное место совпадает	строка от 2 до 255 символов	строка	Формат jpeg	URL на изображение, находящееся в сети интернет	да/нет	строка от 2 до 255 символов	да/нет	строка от 2 до 255 символов	да/нет
Халтурин	Евгений	Александрович	нет	Сибирский федеральный университет	Ассистент института космических информационных технологий	https://disk.yandex.ru/d/Sv91joECbEW6lg	https://disk.yandex.ru/d/WsXoFb2kyPUWrA	https://disk.yandex.ru/d/WsXoFb2kyPUWrA	нет	–	нет	–	да
Малеев	Алексей	Викторович	да	Московский физико-технический институт	Директор Высшей школы программной	https://disk.yandex.ru/d/Sv91joECbEW6lg	https://disk.yandex.ru/d/WsXoFb2kyP	https://disk.yandex.ru/d/WsXoFb2kyPUWrA	нет	–	нет	–	да

					инженерии		UWrA						
Беклемышева	Катерина	Алексеевна	да	Московский физико-технический институт	Доцент кафедры информатики и вычислительной математики	https://disk.yandex.ru/d/Sv91joECbEW6lg	https://disk.yandex.ru/d/WsXoFb2kyPUWrA	https://disk.yandex.ru/d/WsXoFb2kyPUWrA	да	Кандидат физико-математических наук	нет	–	да
Рухович	Филипп	Дмитриевич	да	Московский физико-технический институт	Доцент кафедры алгоритмов и технологий программирования	https://disk.yandex.ru/d/Sv91joECbEW6lg	https://disk.yandex.ru/d/WsXoFb2kyPUWrA	https://disk.yandex.ru/d/WsXoFb2kyPUWrA	да	Кандидат физико-математических наук	нет	–	да
Голубев	Василий	Иванович	Да	Московский физико-технический институт	Доцент кафедры информатики и вычислительной математики	https://disk.yandex.ru/d/Sv91joECbEW6lg	https://disk.yandex.ru/d/WsXoFb2kyPUWrA	https://disk.yandex.ru/d/WsXoFb2kyPUWrA	да	Кандидат физико-математических наук	да	Доцент	да
Васюков	Алексей	Викторович	Да	Московский физико-технический институт	Доцент кафедры информатики и вычислительной	https://disk.yandex.ru/d/Sv91joECbEW6lg	https://disk.yandex.ru/d/WsXoFb2kyPUWrA	https://disk.yandex.ru/d/WsXoFb2kyPUWrA	да	Кандидат физико-математических наук	нет	–	да

					матема тики								

Компетенции

Наименование компетенции	Тип компетенции	Знания, соответствующие компетенции	Умения, соответствующие компетенции	Владение инструментами, соответствующие компетенции
строка, не менее 10 символов	строка, значение из "общекультурные", "общепрофессиональные", "профессиональные"	строка не менее 50 символов, перечень знаний	строка не менее 50 символов, перечень умений	строка не менее 50 символов, перечень инструментов
	Не применимо к дополнительным общеобразовательным программам			

Модули

Наименование поля	Допустимые значения полей	Значение полей	Значение полей	Значение полей	Значение полей
Порядковый номер модуля	целое число	1	2	3	4
Название модуля	строка от 4 символов	Введение в C++	STL и объектно-ориентированное программирование	Проектирование приложений	Оконные и специализированные приложения
Описание модуля	строка до 1000 символов	Модуль включает 8 тем. В конце модуля проводится промежуточная аттестация. При условии успешного окончания модуля выдается сертификат	Модуль включает 8 тем. В конце модуля проводится промежуточная аттестация. При условии успешного окончания модуля выдается сертификат	Модуль включает 6 тем. Предусмотрена подготовка 1 индивидуального промежуточного курсового проекта. В конце модуля проводится	Модуль включает 4 тем. Предусмотрена подготовка 1 группового промежуточного курсового проекта и 1 итогового курсового проекта. В конце

				промежуточная аттестация. При условии успешного окончания модуля выдается сертификат	модуля проводится промежуточная аттестация, а также презентация итоговых курсовых проектов. При условии успешного окончания модуля выдается сертификат
Аттестация по итогам модуля. Количество ак. часов	целое число	4	4	4	8
Аттестация по итогам модуля. Формы контроля	строка не менее 4 символов	Выполнение заданий с автоматизированной проверкой	Выполнение заданий с автоматизированной проверкой	Выполнение заданий с автоматизированной проверкой	Выполнение заданий с автоматизированной проверкой
Аттестация по итогам модуля. Диагностические инструменты	строка не менее 10 символов	Используется автоматизированная проверка выполнения заданий	Используется автоматизированная проверка выполнения заданий	Используется автоматизированная проверка выполнения заданий	Используется автоматизированная проверка выполнения заданий
Аттестация по итогам модуля. Показатели и критерии оценивания	строка не менее 50 символов	- выбор библиотек для решаемой задачи - оптимально / не оптимально (при выборе библиотеки оценивается наличие документации, надёжность, распространённость библиотеки) - качество написания алгоритма решения - оптимально / не оптимально (оценивается скорость работы программы) - работа программы - результативно / нерезультативно (оценивается достижение программой конечного результата,	- выбор библиотек для решаемой задачи - оптимально / не оптимально (при выборе библиотеки оценивается наличие документации, надёжность, распространённость библиотеки) - качество написания алгоритма решения - оптимально / не оптимально (оценивается скорость работы программы) - работа программы - результативно /	- выбор библиотек для решаемой задачи - оптимально / не оптимально (при выборе библиотеки оценивается наличие документации, надёжность, распространённость библиотеки) - качество написания алгоритма решения - оптимально / не оптимально (оценивается скорость работы программы) - работа программы - результативно /	- выбор библиотек для решаемой задачи - оптимально / не оптимально (при выборе библиотеки оценивается наличие документации, надёжность, распространённость библиотеки) - качество написания алгоритма решения - оптимально / не оптимально (оценивается скорость работы программы) - работа программы - результативно /

		заявленного в техническом задании)	нерезультативно (оценивается достижение программой конечного результата, заявленного в техническом задании)	нерезультативно (оценивается достижение программой конечного результата, заявленного в техническом задании)	нерезультативно (оценивается достижение программой конечного результата, заявленного в техническом задании)
Аттестация по итогам модуля. Шкала оценивания, нижнее значение	целое число	0	0	0	0
Аттестация по итогам модуля. Шкала оценивания, верхнее значение	целое число	5	5	5	5
Аттестация по итогам модуля. Шкала оценивания, минимальный проходной балл для успешной сдачи	целое число в диапазоне шкалы	3	3	3	3
Учебно-методические материалы. Методы, формы и технологии	строка не менее 10 символов	Методы обучения: - практическая работа под руководством учителя; - самостоятельная практическая работа; - изучение литературы по теме. Методы контроля: - выполнение практических занятий по темам лекций; - выполнение задания промежуточного контроля. Формы организации учебных занятий:	Методы обучения: - практическая работа под руководством учителя; - самостоятельная практическая работа; - изучение литературы по теме. Методы контроля: - выполнение практических занятий по темам лекций; - выполнение задания промежуточного контроля.	Методы обучения: - практическая работа под руководством учителя; - самостоятельная практическая работа; - изучение литературы по теме. Методы контроля: - выполнение практических занятий по темам лекций; - выполнение задания промежуточного контроля.	Методы обучения: - практическая работа под руководством учителя; - самостоятельная практическая работа; - изучение литературы по теме. Методы контроля: - выполнение практических занятий по темам лекций; - выполнение задания промежуточного контроля.

		- вебинар с элементами практической работы и разбора теоретического материала. Формы организации учебной деятельности: - групповая работа; - индивидуальная работа. Дистанционные образовательные технологии: - использование образовательных интернет-ресурсов; - использование ресурсов, созданных преподавателем (ноутбуки для решения задач по программированию); - WEB-консультации и другие.	Формы организации учебных занятий: - вебинар с элементами практической работы и разбора теоретического материала. Формы организации учебной деятельности: - групповая работа; - индивидуальная работа. Дистанционные образовательные технологии: - использование образовательных интернет-ресурсов; - использование ресурсов, созданных преподавателем (ноутбуки для решения задач по программированию); - WEB-консультации и другие.	Формы организации учебных занятий: - вебинар с элементами практической работы и разбора теоретического материала. Формы организации учебной деятельности: - групповая работа; - индивидуальная работа. Дистанционные образовательные технологии: - использование образовательных интернет-ресурсов; - использование ресурсов, созданных преподавателем (ноутбуки для решения задач по программированию); - WEB-консультации и другие.	Формы организации учебных занятий: - вебинар с элементами практической работы и разбора теоретического материала. Формы организации учебной деятельности: - групповая работа; - индивидуальная работа. Дистанционные образовательные технологии: - использование образовательных интернет-ресурсов; - использование ресурсов, созданных преподавателем (ноутбуки для решения задач по программированию); - WEB-консультации и другие.
Учебно-методические материалы. Методические разработки	строка не менее 10 символов	Не предусмотрено	Не предусмотрено	Не предусмотрено	Не предусмотрено
Учебно-методические материалы. Материалы модуля	строка не менее 10 символов	Примеры заданий: Написать несколько простых программ, на данный момент не обязательно демонстрировать консольный	Примеры заданий: Деструктор, показать его объявление. Продемонстрировать вызов деструктора при	Примеры заданий: Применить навыки ООП для реализации собственной структуры, которой	Примеры заданий: Подготовка к созданию оконных приложений. Установка

		ввод\вывод. Рассмотреть логические и битовые операции. Продемонстрировать различие логических и битовых результатов (например, показав разницу результата при 1&2 и 1&&2). Продемонстрировать способ использования условного оператора if. Дать объяснение оператору фигурные скобки (на данный момент можно условиться, что при помощи фигурных скобок группируются команды в единый выполняемый блок, не указывая на наличие локальной области видимости). Объяснить, что условие if приводится к типу данных bool. Показать возможность применения блока else. Продемонстрировать конструкцию множественного условия switch-case. Объяснить значение команды break и метки default. Продемонстрировать использование команды goto, рассказать, в чём недостаток данной команды и про то, что любой код можно написать без её использования.	помощи отладки, а также для динамических объектов (при помощи new и delete). Показать наиболее частые случаи использования деструктора (для избежания утечек памяти). Рассказать про объявление и назначение константных методов. Продемонстрировать композицию структуры при помощи списка инициализации. Продемонстрировать разбиение структуры на переменные при помощи декомпозиции (стандарт CPP17). Показать при этом, что создаётся копия, а для обращения к оригиналу нужно использовать ссылку. Продемонстрировать, как использовать шаблонный тип данных для структуры и для функций. Показать, какие нюансы возникают при использовании	нет в STL. Продемонстрировать концепцию getter и setter методов. Применить навыки ООП для реализации структуры "Длинные числа". Перегрузить операции. Используем обратный код для выполнения вычитания чисел. Используем дружественные функции для того, чтобы обратиться к приватным полям длинного числа. Реализуем операцию деления и нахождения длинного корня при помощи бинарного поиска. Реализовать умножение через Быстрое преобразование Фурье. Реализовать другую структуру. Возможно, это будет дерево отрезков, возможно, это будет та структура, которая понадобится для итогового проекта (например, структура верстки для написания веб сайта). Применить	подходящей среды разработки и дополнительных библиотек. Установка QT Creator. Настройка компилятора и отладчика. Настройка библиотеки MFC. Использование библиотеки SFML. Использование другой библиотеки, на усмотрение преподавателя. Создание первого оконного проекта. Проверка успешности сборки. Выполнение ранее поставленных задач посредством оконного приложения (преобразование ранее написанных консольных приложений в оконные).

			шаблонов (например, несовпадения типа данных аргументов и невозможность их выбора, как в std::min при разных аргументах). Показать, что такое тип auto и как он может быть использован в описании функции (для параметров и для результирующего значения). Указать на то, что шаблонные структуры не могут иметь реализацию в другом файле, где нет объявления. Показать, как возможно отделить объявление от реализации для шаблонных структур. Показать, как возможно использовать intelliSense для указания явных типов у шаблона.	описание дружественных функций. Показать, как осуществить ассемблерную вставку. Рассмотреть возможную реализацию при помощи сторонней библиотеки (возможно библиотеки boost). Показать, как применять модульное тестирование.	
Учебно-методические материалы. Учебная литература	строка не менее 10 символов	Поляков К. Ю., Еремин Е. А. Информатика. Углублённый уровень. Учебник для 10 класса в 2 частях. М.: БИНОМ. Лаборатория знаний, 2014.	Поляков К. Ю., Еремин Е. А. Информатика. Углублённый уровень. Учебник для 10 класса в 2 частях. М.:	Поляков К. Ю., Еремин Е. А. Информатика. Углублённый уровень. Учебник для 10 класса в 2 частях. М.:	Поляков К. Ю., Еремин Е. А. Информатика. Углублённый уровень. Учебник для 10 класса в 2 частях. М.:

		Информатика и ИКТ. Задачник-практикум в 2 частях. Под ред. И. Г. Семакина и Е. К. Хеннера. М.: БИНОМ. Лаборатория знаний, 2014. Лааксонен А. Олимпиадное программирование: ДМК Пресс, 2022. Мартин Р. Идеальный программист. Как стать профессионалом разработки ПО: СПб. Питер, 2011. Гамма Э. и др. Приемы объектно-ориентированного проектирования. Паттерны проектирования: Учебное пособие, 2007. Халим С., Халим Ф. Спортивное программирование: ДМК Пресс, 2022 Кнут Д. Э. Искусство программирования: Издательский дом Вильямс, 2000.	БИНОМ. Лаборатория знаний, 2014. Информатика и ИКТ. Задачник-практикум в 2 частях. Под ред. И. Г. Семакина и Е. К. Хеннера. М.: БИНОМ. Лаборатория знаний, 2014. Лааксонен А. Олимпиадное программирование: ДМК Пресс, 2022. Мартин Р. Идеальный программист. Как стать профессионалом разработки ПО: СПб. Питер, 2011. Гамма Э. и др. Приемы объектно-ориентированного проектирования. Паттерны проектирования: Учебное пособие, 2007. Халим С., Халим Ф. Спортивное программирование: ДМК Пресс, 2022 Кнут Д. Э. Искусство программирования: Издательский дом Вильямс, 2000.	БИНОМ. Лаборатория знаний, 2014. Информатика и ИКТ. Задачник-практикум в 2 частях. Под ред. И. Г. Семакина и Е. К. Хеннера. М.: БИНОМ. Лаборатория знаний, 2014. Лааксонен А. Олимпиадное программирование: ДМК Пресс, 2022. Мартин Р. Идеальный программист. Как стать профессионалом разработки ПО: СПб. Питер, 2011. Гамма Э. и др. Приемы объектно-ориентированного проектирования. Паттерны проектирования: Учебное пособие, 2007. Халим С., Халим Ф. Спортивное программирование: ДМК Пресс, 2022 Кнут Д. Э. Искусство программирования: Издательский дом Вильямс, 2000.	БИНОМ. Лаборатория знаний, 2014. Информатика и ИКТ. Задачник-практикум в 2 частях. Под ред. И. Г. Семакина и Е. К. Хеннера. М.: БИНОМ. Лаборатория знаний, 2014. Лааксонен А. Олимпиадное программирование: ДМК Пресс, 2022. Мартин Р. Идеальный программист. Как стать профессионалом разработки ПО: СПб. Питер, 2011. Гамма Э. и др. Приемы объектно-ориентированного проектирования. Паттерны проектирования: Учебное пособие, 2007. Халим С., Халим Ф. Спортивное программирование: ДМК Пресс, 2022 Кнут Д. Э. Искусство программирования: Издательский дом Вильямс, 2000.

Материально-технические условия реализации программы. Наименование требуемого оборудования	строка не менее 2 символов	Персональные компьютеры/ноутбуки, среда разработки программного обеспечения, доступ в Интернет	Персональные компьютеры/ноутбуки, среда разработки программного обеспечения, доступ в Интернет	Персональные компьютеры/ноутбуки, среда разработки программного обеспечения, доступ в Интернет	Персональные компьютеры/ноутбуки, среда разработки программного обеспечения, доступ в Интернет
Материально-технические условия реализации программы. Наименование требуемого программного обеспечения	строка не менее 2 символов	– операционная система (желательно Windows); – браузер: Yandex Browser, Chrome, Chrome Mobile, Firefox, Opera, Safari, Mobile Safari, Edge; – среда разработки программного обеспечения: Microsoft Visual Studio, Code::blocks, CLion, Visual Studio Code.	– операционная система (желательно Windows); – браузер: Yandex Browser, Chrome, Chrome Mobile, Firefox, Opera, Safari, Mobile Safari, Edge; – среда разработки программного обеспечения: Microsoft Visual Studio, Code::blocks, CLion, Visual Studio Code.	– операционная система (желательно Windows); – браузер: Yandex Browser, Chrome, Chrome Mobile, Firefox, Opera, Safari, Mobile Safari, Edge; – среда разработки программного обеспечения: Microsoft Visual Studio, Code::blocks, CLion, Visual Studio Code.	– операционная система (желательно Windows); – браузер: Yandex Browser, Chrome, Chrome Mobile, Firefox, Opera, Safari, Mobile Safari, Edge; – среда разработки программного обеспечения: Microsoft Visual Studio, Code::blocks, CLion, Visual Studio Code.
Электронные информационные ресурсы	строка не менее 10 символов				
Электронные образовательные ресурсы	строка не менее 10 символов	Мирзаянов М. Платформа для соревнований по программированию Codeforces. 2013. [Электронный ресурс] URL: https://codeforces.com/ (дата обращения: 10.08.2022). Престон-Вернер Т. И др. Веб-сервис для хостинга IT-проектов и их совместной разработки GitHub. 2008. [Электронный ресурс] URL: https://github.com/ (дата обращения: 10.08.2022).	Мирзаянов М. Платформа для соревнований по программированию Codeforces. 2013. [Электронный ресурс] URL: https://codeforces.com/ (дата обращения: 10.08.2022). Престон-Вернер Т. И др. Веб-сервис для хостинга IT-проектов	Мирзаянов М. Платформа для соревнований по программированию Codeforces. 2013. [Электронный ресурс] URL: https://codeforces.com/ (дата обращения: 10.08.2022). Престон-Вернер Т. И др. Веб-сервис для хостинга IT-проектов	Мирзаянов М. Платформа для соревнований по программированию Codeforces. 2013. [Электронный ресурс] URL: https://codeforces.com/ (дата обращения: 10.08.2022). Престон-Вернер Т. И др. Веб-сервис для хостинга IT-проектов

		Спольски Д. Визуальный веб-инструмент управления проектами, рабочими процессами и заданиям Trello. 2011. [Электронный ресурс] URL: https://trello.com/ (дата обращения: 10.08.2022).	и их совместной разработки GitHub. 2008. [Электронный ресурс] URL: https://github.com/ (дата обращения: 10.08.2022). Спольски Д. Визуальный веб-инструмент управления проектами, рабочими процессами и заданиям Trello. 2011. [Электронный ресурс] URL: https://trello.com/ (дата обращения: 10.08.2022).	и их совместной разработки GitHub. 2008. [Электронный ресурс] URL: https://github.com/ (дата обращения: 10.08.2022). Спольски Д. Визуальный веб-инструмент управления проектами, рабочими процессами и заданиям Trello. 2011. [Электронный ресурс] URL: https://trello.com/ (дата обращения: 10.08.2022).	и их совместной разработки GitHub. 2008. [Электронный ресурс] URL: https://github.com/ (дата обращения: 10.08.2022). Спольски Д. Визуальный веб-инструмент управления проектами, рабочими процессами и заданиям Trello. 2011. [Электронный ресурс] URL: https://trello.com/ (дата обращения: 10.08.2022).

Темы модулей

Порядковый номер модуля	Наименование темы	Лекции. Количество академических часов	Содержание лекций	Практические занятия. Количество академических часов	Содержание практических занятий	Самостоятельная работа. Количество академических часов	Содержание самостоятельной работы
Целое число 1..4	строка от 4 до 255 символов	Целое число часов	строка не менее 50 символов при ненулевом количестве академических часов лекций	Целое число часов	строка не менее 100 символов при ненулевом количестве академических часов лекций	Целое число часов	строка не менее 50 символов при ненулевом количестве академических часов лекций
1	Введение. Достоинства и			2	Введение. Достоинства и недостатки языка С++.	2	Самостоятельное выполнение

	недостатки языка C++. Установка среды разработки. Типы переменных и арифметические операции				Установка среды разработки. Типы переменных и арифметические операции Как и для каких целей появился язык C++, какая связь языка C++ и чистого СИ. Рассказ о том, в каких индустриях применяется язык C++ (можно упомянуть Unreal Engine и разработку компьютерных игр), а также о том, насколько эффективными получаются итоговые программные решения. Но, также стоит объяснить и о том, насколько сложно вести разработки на языке C++. Установка среды разработки Visual Studio, создание первого консольного приложения. После этого объясняется, что из себя представляет наша программа и из каких основных частей состоит программный код. Установка другой IDE, установка компилятора и отладчика, указание путей по умолчанию, связывание IDE со вспомогательными средствами. Продемонстрировать возможности отладчика,		тренировочных заданий по теме

					пошаговое прохождение программы. использование точек остановки. Перечислить преимущества установленной среды разработки от онлайн аналогов (но при этом указав их, как альтернативы, возможные варианты: компилятор codeforces, GDB online Debugger и другие). Перечислить возможные типы переменных (не забыть про unsigned), их различия, способ записи данных в битах, размеры. Продемонстрировать арифметические операции, объяснить их способ использования. Указать на то, что есть быстрые и медленные операции.		
1	Логические и битовые операции. Разветвления хода выполнения программы			2	Написать несколько простых программ, на данный момент не обязательно демонстрировать консольный ввод\вывод. Рассмотреть логические и битовые операции. Продемонстрировать различие логических и битовых результатов (например, показав разницу результата при 1&2 и 1&&2).	2	Самостоятельное выполнение тренировочных заданий по теме

					Продемонстрировать способ использования условного оператора if. Дать объяснение оператору фигурные скобки (на данный момент можно условиться, что при помощи фигурных скобок группируются команды в единый выполняемый блок, не указывая на наличие локальной области видимости). Объяснить, что условие if приводится к типу данных bool. Показать возможность применения блока else. Продемонстрировать конструкцию множественного условия switch-case. Объяснить значение команды break и метки default. Продемонстрировать использование команды goto, рассказать, в чём недостаток данной команды и про то, что любой код можно написать без её использования.		
1	Приведение типов. Указатели и ссылки. Оператор запятая. Приоритет операций. Ввод и вывод данных			3	Рассказать про преобразование типов данных. Указать на то, что константам можно явно указывать тип данных при помощи соответствующего суффикса. Указать	1	Самостоятельное выполнение тренировочных заданий по теме

					приоритет и направление выполнения операций. Продемонстрировать различие префиксного и суффиксного инкремента. Показать, как выбирается результирующий тип данных, а также возможные ситуации переполнения. Показать сложно составленные строки программного кода с несколькими действиями, задать вопросы на понимание, что получится в результате выполнения. Указать на отличие одинарной кавычки от двойной. Показать тернарное условие и его основное применение (для вывода результирующего значения, а не для выполнения действий). Рассказать про то, что у переменных имеются уникальные адреса. Абстрактно объяснить, как хранятся данные о переменных в памяти. Продемонстрировать работу с указателями. Показать операции получения адреса и получения объекта по адресу. Показать возможность создание		

					псевдонимов посредствам ссылок, а также то, как от указателей перейти к ссылкам. Показать преобразование типов данных посредством указателей, а также объяснить различие преобразования посредством оператора присваивания (то есть отличие от того, как преобразовывалось ранее). Продемонстрировать использование оператора запятой. Показать, как выводится результирующее значения при применении оператора запятая. Показать способ ввода и вывода данных в программу. При помощи библиотеки iostream и консольного потока ввода\вывода cin\cout. Объяснить, что cin\cout - это сложно устроенные объекты, служащие буфером хранения данных из источника (в нашем случае это консоль) для ввода и параллельно для вывода. При помощи scanf printf и библиотеки cstdio (или stdio.h в зависимости от компилятора). Рассказать,		

					что scanf и printf - это функции, а что такое функции будет объясняться на следующих лекциях. Также отметить, что данные функции имеют неограниченное количество входных параметров		
1	Подробно про библиотеки. Циклы, 3 формы, их сравнение. Точки останова по условия. Вложенные циклы			2	Перечислить основные подключаемые библиотеки через #include и кратко дать их объяснение, чем они могут быть полезны. Объяснить, что все команды, начинающиеся через # являются мета-командами. Показать, что такое макросы #define и условная компиляция #ifdef. Реализовать циклы при помощи for, префиксного и постфиксного while. Научить оценивать асимптотику программы по времени и по памяти. Реализовать вложенные циклы, спросить у учащихся их асимптотики. Показать возможный вариант цикла с асимптотикой O(log(n)). Объяснить, почему в асимптотики у логарифма не ставится основание (из-за формулы перехода к другому основанию, показать её наличие). Сравнить асимптотики, дать	2	Самостоятельное выполнение тренировочных заданий по теме

					оценку выполнения программы за 1 секунду (при каких входных данных успеет завершиться программа за 1 секунду с заданой асимптотикой). Рассказать про оператор continue и способ написания циклов с выходом через условия. Показать пример асимптотики O(sqrt(n)) (это может быть задача определения простоты перебором делителей до корня).		
1	Время жизни переменных. Глобальные переменные. Динамическое выделение памяти. Массивы. Многомерные массивы			2	Рассказать про области видимости, задающиеся фигурные скобками, а также про время жизни статичных переменных (тех, что удаляются из памяти автоматически, когда закрывается их область видимости). Рассказать про глобальные переменные и их начальную инициализацию. Рассказать про оператора new и динамическое выделение памяти, а также о том, как отличается время жизни динамических переменных от динамических.	2	Самостоятельное выполнение тренировочных заданий по теме

					Другие способы выделения памяти, например про функции из чистого СИ. Рассказать, что такое оператор sizeof. Рассказать, что такое утечки памяти и какой вред они могут нести. Научить отлавливать утечки памяти с помощью библиотеки CRT (<crtdbg.h>) Рассказать про команду delete. Показать, как объявлять статичные массивы (на фиксированное количество элементов). Поговорить про то, как выбирать удачнее всего размер статичного массива, исходя из поставленной задачи. Показать, как объявлять многомерные массивы. Применить их в программе совместно с циклами и условиями. Показать, как создавать динамический массив при помощи new и указателя. Рассказать про особенность использования char*, а также про то, какие ньюансы могут возникнуть при отладке (отладчик воспринимает указатель на char, как строку, ограниченную '\0').		

					Показать, насколько сложнее работать с динамическими многомерными массивами (потребуются циклы для объявления). Рассказать про преимущества использования динамических многомерных массивов во многих задачах (за счёт ступенчатого вида, а непрямоугольного, как в случае статических). Продемонстрировать возможный вариант изменения размера динамического массива и переноса данных. Показать, что оператор квадратные скобки есть смещение по адресу, хранимом в указателе. Показать упрощённый вариант использования динамических массивов посредством std::vector.		
1	Вспомогательные программы программиста, репозиторий. Функции и процедуры. Способы передачи аргументов в функцию			3	Показать возможный способ введения, планирования, проектирования и хранения проекта. Сохранить в общий репозиторий весь ранее написанный код для свободного доступа к нему ученикам. Объяснить, что такое функция, на примере	1	Самостоятельное выполнение тренировочных заданий по теме

					простых действий (например, написав функцию суммирования или определения принадлежности символа заглавным). Указать на синтаксис описания функции. Рассказать про отличительную особенность функции main от остальных, а также о том, в чём смысл возвращаемого ею значения и аргументов. Напомнить про приведение типов при вызове функций. Показать, как применимы указатели при взаимодействии с функцией, чтобы изменять внешние для неё переменные. Показать, как реализовать перегрузку функции. Объяснить, какая перегрузка применима и почему (та, что различима однозначно компилятором). Указать на использование значений по умолчанию для параметров. Поговорить про константные переменные. Показать, как передать оригинал в функцию посредством ссылки. Рассказать про копирование данных в переменные под аргументы, если не использовать ссылки.		

					Показать, как сочетается константность и ссылка и зачем это используется на практике (чтобы не выделять новую память, но при этом, чтобы нельзя было изменить значение переменной).		
1	Пространство имён. Рекурсия. Графы			2	Показать, как объявляется пространство. Объяснить, зачем оно используется. Рассказать про то, что основной функционал языка C++ находится в пространстве имён std. Показать обозреватель объектов Visual Studio и то, как пространства имен упаковывают внутрь себя функции и глобальные переменные. Объяснить на примерах, что такое рекурсия, где она встречается в повседневности (например, рекурсией оперирует файловая система, так в папке могут находиться другие папки). Показать, как через рекурсию можно переписать циклы. Показать, как применить рекурсию для задач на перебор состояний и для других комбинаторных задач. Указать на то, что данные мы вводим слева-	2	Самостоятельное выполнение тренировочных заданий по теме

					направо и снизу вниз, но при помощи рекурсии обработку данных можно изменить на сверху-вниз и слева-направо. Рассказать про то, что такое стек и что такое стек вызова функций. Изменить размер стека функций при помощи команды компилятору pragma comment(linker, "/STACK: "). Пройтись в отладчике по программе, использующей рекурсию. Рассказать, что такое граф, как его можно хранить (3 формы представления) и как совершить по нему поиск в глубину при помощи рекурсии. Совершить поиск пути в неявном графе (лабиринте) при помощи рекурсии.		
1	Оптимизация рекурсия. Поиск ответа перебором. Использование подхода динамического программирования. Работа с файлами			2	Показать, как при помощи возвращаемого значения возможно остановить поиск при помощи рекурсии. Показать, как возможно восстановить ответ при помощи рекурсии. Рассказать, что такое динамическое программирование. Показать примеры задач, например, нахождение чисел Фибоначчи или для задач подсчёта разбиений	2	Самостоятельное выполнение тренировочных заданий по теме

					по указанному правилу. Совершить оптимизацию при помощи ленивой реализации динамического программирования. Показать, как возможно считать и вывести данные в файл. Показать перенаправление консольного потока на файл при помощи функции freopen. Показать защищенный вариант функции freopen_s, а также то, что такое проверка SDL Visual Studio, как её отключить и\или как добавить отключение проверок на небезопасные функции (при помощи #pragma warning(disable : 4996)). Рассказать про файловые потоки и библиотеку <fstream>. Рассказать про преимущества данного подхода работы данных в отличии от перенаправления потока. Показать другие способы работы с файлами, например, через указатели на файл. Рассказать про положительные и отрицательные стороны применения рекурсии.		

					Рассказать про синхронизацию старого ввода\вывода с новым. Отключить ей при помощи ios_base sync_with_stdio(false). Отключить синхронизацию ввода и вывода при помощи cin.tie(false), чтобы ускорить ввод и вывод данных.		
2	Структуры, её поля и методы. Оперирование объектами. Разбиение проекта на объявление и реализацию			2	Краткий рассказ о том, что из себя представляет объектно-ориентированное программирование, где используется оно, но также пояснить и то, где используется функциональное программирование. Показать синтаксис объявления структур, пока показывать только наличие полей. Указать на то, что теперь программа оперирует объектами, которые содержат совокупность полей. Написать несколько программ с использованием структур. Переписать предыдущие программы с использованием структур. Показать, как передавать структуры в функцию и что произойдёт при использовании глобальных	2	Самостоятельное выполнение тренировочных заданий по теме

					структур, какова их начальная инициализация. Показать что такое методы. Продемонстрировать их ассоциацию с объектом. Показать, что возможно динамически создавать структуры при помощи new. Показать, что такое оператор стрелочка. Разбить проект на файлы .cpp и .h, рассказать про пользу отделения описания от реализации.		
2	Конструктор и деструктор. Константные методы. Композиция, декомпозиция и лист инициализации. Шаблонный тип данных. auto			3	Рассказать, что такое конструктор, как его объявить. Рассказать про конструктор по умолчанию и его перекрытие собственным конструктором. Рассказать про то, что для функций (но не для методов) без параметров не требуется указывать круглые скобки. Продемонстрировать вызов конструктора при помощи отладки. Деструктор, показать его объявление. Продемонстрировать вызов деструктора при помощи отладки, а также для динамических объектов (при помощи new и delete). Показать наиболее частые случаи использования	1	Самостоятельное выполнение тренировочных заданий по теме

					деструктора (для избежания утечек памяти). Рассказать про объявление и назначение константных методов. Продемонстрировать композицию структуры при помощи списка инициализации. Продемонстрировать разбиение структуры на переменные при помощи декомпозиции (стандарт CPP17). Показать при этом, что создаётся копия, а для обращения к оригиналу нужно использовать ссылку. Продемонстрировать, как использовать шаблонный тип данных для структуры и для функций. Показать, какие нюансы возникают при использовании шаблонов (например, несовпадения типа данных аргументов и невозможность их выбора, как в std::min при разных аргументах). Показать, что такое тип auto и как он может быть использован в описании функции (для параметров и для результирующего значения). Указать на то, что шаблонные структуры не		

					могут иметь реализацию в другом файле, где нет объявления. Рассказать про изменения, которые затронули auto в различных стандартах языка. Рассказать более подробно про список инициализации, его блокировку приведения типов и использование при описании конструктора, а также, какие плюсы возникают при его использовании в конструкторе. Показать, как возможно отделить объявление от реализации для шаблонных структур. Показать, как возможно использовать intelliSense для указания явных типов у шаблона.		
2	Перегрузка операторов. Функции, как параметры. std::vector, std::string, std::stack			2	Показать синтаксис перегрузки операторов. Показать набор возможных перегрузок. Продемонстрировать для своих структур. Рассказать, что такое void*. Показать, как возможно передать фунцию, как параметр. При помощи библиотеки <functional> и структуры std::function. Продемонстрировать	2	Самостоятельное выполнение тренировочных заданий по теме

					примерами. При помощи стандартного синтаксиса и при помощи void*. Показать, что vector является шаблонной структурой с перегруженными операторами. Реализовать собственный vector (с конструктором и деструктором). Показать методы vector, а также string и stack. Написать перегрузку оператора >> для считывания vector при помощи стандартных потоков. Показать, как возможно считать данные построчно, посимвольно и до конца файла. Показать, как string преобразовать в поток std::stringstream и записать из него данные в переменные. показать возможности библиотеки <iomanip> и вывода вещественного числа с указанной точностью (а также как точность указывается в scanf).		
2	std::list, итераторы, поэлементный обход. Собственная реализация			2	Объяснить различие std::list от std::vector, завести таблицу с указанием скорости выполнения операций для разных структур данных. Показать, как list хранится в памяти.	2	Самостоятельное выполнение тренировочных заданий по теме

					Итераторы, объяснить их назначение. Рассказать про виды итераторов и про то, как ими пользоваться в программе. Показать вставку по итератору, и что происходит при удалении по итератору. Обратные итераторы (и как они конвертируются друг в друга). Показать поэлементный обход по структурам. Показать различие копирования данных в поэлементном обходе, от обращения по ссылки. Продемонстрировать декомпозицию в поэлементном обходе. Осуществить собственную реализацию двусвязного (или односвязного) списка. Реализовать свои итераторы. Показать функции std::next, std::prev, std::advance и std::distance.		
2	Пары и кортежи. std::queue, std::deque. Сортировка. Компаратор			2	Показать возможность применения нескольких шаблонных типов, а также как оперировать с std::pair. Показать, как использовать std::tuple и функцию std::get. Рассказать про структуры	2	Самостоятельное выполнение тренировочных заданий по теме

					данных std::queue и std::deque. Оценить время выполнения операций. Рассказать, как хранится в памяти std::deque. Сделать собственную реализацию. Способы сортировки, начиная с медленных (O(n^2)). Рассказать про сортировку подсчётом. Рассказать про быстрые сортировки (O(n*log(n)), например про сортировку слиянием. Реализовать быструю сортировку (разделяй-и-властвуй, она же Хоара). Возможные оптимизации сортировки, например с использованием троичного разбиения и выбора случайного опорного элемента. Для простых сортировок можно отметить, когда возможно досрочное прекращение. Показать встроенные в STL сортировки (std::sort и другие, зависит от компилятора). Показать, что такое компаратор-функция и что такое компаратор-структура. Применить компаратор для std::sort (упомянуть про std::less и std::greater).		

2	Библиотека <algorithm>. Бинарный поиск. Перестановка. Передача неограниченного числа параметров			2	Показать реализованные функции в библиотеки <algorithm>. Объяснить, что такое бинарный поиск, как он реализуется. Показать левосторонний и правосторонний бинарный поиск, показать их отличие (когда есть одинаковые элементы выбираются разные крайние позиции). Показать бинарный поиск по ответу (то есть подбор варианта, но с сокращением пространства поиска в два раза). Показать, как использовать std::lower_bound и std::upper_bound, их отличия друг от друга, когда функции применимы. Рассказать, что такое тернарный (троичный) поиск. Возможно, привести пример рычажных весов и поиска фальшивой монеты. Рассказать, что такое перестановка. Воспользоваться функцией next_permutation для поиска следующей перестановки. Рассказать, что такое constexpr и зачем он применяется.	2	Самостоятельное выполнение тренировочных заданий по теме

					Реализовать собственную функцию генерации перестановок (или получения соседней). Передача неограниченного количества аргументов в функцию, например, при помощи va_list. Метапрограммирование и предподсчёт на этапе компиляции. Передача неограниченного количества шаблонных типов.		
2	Бинарное дерево. Бор. std::set, std::multiset, std::map, std::bitset			3	Построить бинарное дерево поиска (следуя правилу, что меньший стоит добавлять слева, больший справа, с дубликатами как обращаться решает преподаватель). Показать, как его можно использовать и какие проблемы возникают при добавлении и удалении элементов. Показать дерево Бор, решить некоторые простые задачи с его использованием (например, хранение телефонной книги). Рассказать про структуру данных std::set, её методы, указать в общей таблице асимптотику выполнения операций.	1	Самостоятельное выполнение тренировочных заданий по теме

					Рассказать, что такое красно-чёрное (или АВЛ) дерево, его свойства, примерно указать, как реализуются команды добавления и удаления в дерево (рассказать про наличие поворотов). Передать компаратор в std::set. Продемонстрировать решение задач при помощи std::set. Рассказать про наличие std::multiset, его сходство и то, как альтернативно можно было реализовать multiset через обычный set (с использованием pair). Рассказать про итераторы set и про запрет на изменение значений напрямую. Рассказать про std::map, сравнить его с set, показать добавочные операции (перегруженные квадратные скобки) и как его применяют. Решить задачи с его использованием. Сделать поэлементный обход с возможностью изменения значений. Добавить компаратор. Указать ситуации, когда set и map не удастся		

					использовать (есть не сравниваемые оператором меньше собственные структуры). Использовать set и map для ранее написанных собственных структур. Указать в общей таблице (где сравниваются структуры данных STL) асимптотику выполнения разобранных структур данных. Рассказать про bitset, то, как можно использовать переменные для тех же задач, как инвертировать определённые биты.		
2	std::priority_queue. Случайные значения. Простые числа. std::unordered_map. Хеширование			2	Рассказать про структуру данных std::priority_queue, ей методы, когда используется. Также рассказать про функцию make_heap. Сравнить скорости выполнения set и priority_queue для одной и той же задачи (например, для алгоритма Дейкстры). Для измерения расходов памяти и требуемого времени можно воспользоваться средством диагностики Visual Studio. Рассказать, что такое структура данных куча, ей свойства, как она устроена.	2	Самостоятельное выполнение тренировочных заданий по теме

					Возможно упомянуть в том числе про Декартово дерево. Рассказать про функцию rand и функцию srand, а также про возможность получения текущего времени при помощи функции time. Рассказать про библиотеку <random> и про используемые в ней генераторы (например, для получения long long). Рассказать, что представляют из себя простые числа. Предложить алгоритм нахождения простых чисел, а также алгоритм проверки за простоту (желательно с асимптотикой O(sqrt(n)). Рассказать про Китайскую теорему об остатках и её конструктивное применение. Рассказать про std::unordered_map, её методы и для каких данных применима (только для тех, что имеют алгоритм хеширования). Рассказать про асимптотику и про скрытую константу, а также про случаи, когда unordered_map эффективнее		

					map. Рассказать про коллизии и про опасность увеличения асимптотики с O(log(n)) до O(n). Рассказать про алгоритм хеширования и про то, как устроены хеш-таблицы. Сделать собственную реализацию одним из способов. Полиномиальный хеш и когда он применим, для каких задач.		
3	Модификатор доступа. Наследование. Виртуальные методы. Абстрактный класс. Ассоциация объектов			3	Построить концепцию ООП с учащимися. Рассказать про модификаторы доступа, в чем отличие protected от private. Рассказать, в чём отличие класса от структур (в модификаторе доступа, но можно упомянуть о различиях, которые были в чистом СИ). Продемонстрировать наследование, рассказать, когда оно применяется. Продемонстрировать мультинаследование и рассказать, какой от него вред, какие проблемы проектирования могут возникнуть при мультинаследовании. Рассказать про виртуальные методы, в чём их назначение. Рассказать про	2	Самостоятельное выполнение тренировочных заданий по теме

					Абстрактные классы и чисто виртуальные методы. Показать различие между композицией и агрегацией при проектировании проекта. Показать UML диаграммы. Возможно, продемонстрировать построение UML диаграмм при помощи Microsoft Visio. Рассказать про известные ООП проекты и их архитектуры, UML диаграммы.		
3	Применения навыков ООП. Дружественные функции. Модульное тестирование			3	Применить навыки ООП для реализации собственной структуры, которой нет в STL. Продемонстрировать концепцию getter и setter методов. Применить навыки ООП для реализации структуры "Длинные числа". Перегрузить операции. Используем обратный код для выполнения вычитания чисел. Используем дружественные функции для того, чтобы обратиться к приватным полям длинного числа. Реализуем операцию деления и нахождения длинного корня при помощи бинарного поиска. Реализовать умножение через Быстрое преобразование Фурье.	1	Самостоятельное выполнение тренировочных заданий по теме

					Реализовать другую структуру. Возможно, это будет дерево отрезков, возможно, это будет та структура, которая понадобится для итогового проекта (например, структура верстки для написания веб сайта). Применить описание дружественных функций. Показать, как осуществить ассемблерную вставку. Рассмотреть возможную реализацию при помощи сторонней библиотеки (возможно библиотеки boost). Показать, как применять модульное тестирование.		
3	Работа с файловой системой и внешними ресурсами. Безопасные указатели. Исключения			3	Воспользоваться библиотекой <filesystem> для решения прикладных задач. Осуществить работу с несколькими файлами. Это могут быть ресурсы, которые будут использованы для итогового проекта. В случае компьютерной игры это могут быть изображения и другие медиа ресурсы, в случае веб сайта это могут быть html страницы. Для научной таблицы это могут быть таблицы в формате .csv. Применить умные	2	Самостоятельное выполнение тренировочных заданий по теме

					указатели (библиотека <memory>). Показать отличия между std::unique_ptr , std::shared_ptr , std::weak_ptr. Продемонстрировать отсутствие утечек памяти. Рассказать про исключения, библиотеку <stdexcept> конструкцию try catch. Обратить внимание на то, что процесс выполнения не прерывается командой throw, если нет внешнего try. Рассказать про модификатор функции noexcept. Рассказать про библиотеку <typeinfo> и определения типа передаваемого параметра (может оказаться полезным в случае шаблонной функции или использования типа auto).		
3	Многопоточность и многопроцессорность. Гонка ресурсов			3	Создание многопоточных приложений. Использовать библиотеку <thread>. Импортировать библиотеку boost. Рассказать про гонку ресурсов, мьютексы и семафоры. Показать их схожесть с умными указателями. Рассказать, в чём отличие многопотоковых от многопроцессорных	2	Самостоятельное выполнение тренировочных заданий по теме

					приложений. Создание многопроцессорного приложения с использованием любой удобной библиотеки (рекомендуется OpenMP). Показать передачу данных в многопроцессорных приложениях при помощи пересылки сообщений.		
3	Применение параллельных вычислений. Работа с ядрами видеокарты. Матричные операции над варпами, блоками и сетками			3	Продолжение объяснения материала параллельного программирования. Демонстрация возможностей многопроцессорных программ. Составление программ, обрабатывающих запросы параллельно. Продолжения изучения материала предыдущего занятия. Рассмотрение возможности выполнения операций с использованием видеокарты. Использование библиотеки для работы с CUDA ядрами. Объяснение нюансов использования ресурсов видеокарты, специфичности обработки данных и применяемых операций (объяснить, что выполняются операции над многомерными матрицами большой размерности, под каждый варп выделяется отдельный поток, поэтому	2	Самостоятельное выполнение тренировочных заданий по теме

					операции над каждым элементом происходят синхронно).		
3	Работа в команде, распределение обязанностей. Управление временем			3	Время проектирования и жизнь проекта, какие этапы развития проходит проект. Занятие строится на опыте преподавателя. Рассказать про средства совместной разработки, про общий репозиторий и его разветвление. Рассмотреть механизм слияния ветвей в общем репозитории. Основные обязанности в команде для реализации проекта заданной итоговой работой направленности (если это реализация компьютерной игры, то это дизайнер, аниматор, проектировщик внутренней логики, интерфейс-дизайнер и другие роли, если это разработка, то это Frontend и Backend разработчик).	1	Самостоятельное выполнение тренировочных заданий по теме
3	***Индивидуальный промежуточный курсовой проект***					***4***	***Подготовка индивидуального промежуточного курсового проекта***
4	Разработка оконных приложений			4	Подготовка к созданию оконных приложений. Установка подходящей среды разработки и дополнительных библиотек. Установка QT Creator. Настройка компилятора и	1	Самостоятельное выполнение тренировочных заданий по теме

					отладчика. Настройка библиотеки MFC. Использование библиотеки SFML. Использование другой библиотеки, на усмотрение преподавателя. Создание первого оконного проекта. Проверка успешности сборки. Рассказ об аналогии создания оконных приложений на других языках программирования, например на C#. Объяснение разделения интерфейса пользователя от логики проекта. Выполнение ранее поставленных задач посредством оконного приложения (преобразование ранее написанных консольных приложений в оконные).		
4	Продолжение работы с оконными приложениями. Разделение приоритетов, распределение ролей в команде и составление плана работ			5	Разработка оконных приложений. Рассмотрение часто встречаемых ошибок. Управление оконными свойствами. Рассмотрение набора оконных элементов. Разбор паттернов разработки: события, делегаты, фабричный метод, фасад. Определить цель и задачи последующего курсового проекта, определить то, что должна	1	Самостоятельное выполнение тренировочных заданий по теме

					выполнять оконная программа по итогу выполнения. Распределение учащихся по командам, распределение их обязанностей. Установление сроков выполнения (желательно с указанием 3 оценок временных интервалов по схеме PERT, но возможно и другие способы, известные преподавателю).		
4	Специализированное приложение			4	Определить новую среду разработки или библиотеку для реализации итогового курсового проекта. В зависимости от итогового курсового проекта и специализации преподавателя. Это может быть среда или библиотека для разработки графических приложений, таких как компьютерных игр или построителей графиков. Возможно это будет среда для разработки веб-приложения, мобильного приложения, высокопроизводительного приложения для научных вычислений или разработки искусственного интеллекта. Как вариант, средой разработки может являться Unreal Engine или возможно	1	Самостоятельное выполнение тренировочных заданий по теме

					потребуется библиотека SFML. Возможен вариант подключения библиотеки для работы с базой данных. Показать учащимся, как взаимодействовать с новой средой разработки и как разрабатывать в ней приложение.		
4	Разработка приложения с учётом специфики			5	В зависимости от выбранной специфики: показать возможности встроенных функций. Если это новая среда разработки, показать встроенный функционал, вывести наиболее часто используемые в работе окна обозревателя. Показать, как найти и пользоваться документацией. Разобрать возможные гибкие решения, для упрощения дальнейшей модификации проектов. Распределить учащихся по командам или дать индивидуальные задания (возможен смешанный вариант). Предложить темы итогового курсового проекта.	1	Самостоятельное выполнение тренировочных заданий по теме
4	***Командный промежуточный курсовой проект***					***2***	***Подготовка командного промежуточного курсового проекта***

4	***Итоговый курсовой проект***					***4***	***Подготовка итогового курсового проекта***

Адреса и координаты

№ п/п	Название адреса	Адрес	Код адреса	Долгота	Широта
целое число	строка	строка	Целое число	вещественное число	вещественное число
1	Федеральное государственное автономное образовательное учреждение высшего образования «Московский физико-технический институт (национальный исследовательский университет)»	141701, Московская обл., г. Долгопрудный, Институтский пер., д. 9, стр.3	46416000000	55.929444	37.518434
2	МОУ «Лицей № 1»	185014, Республика Карелия, Петрозаводск, Березовая аллея, 42 (р-н Древлянка)	86401000000	61.768441	34.310532
3	ГБОУ города Москвы «Школа № 2006»	Москва, улица Грина, дом 18, корпус 3, строение 1	45293582000	55.565768	37.582367
4	ГБОУ «Школа №2107»	129110, Москва, Большая Переяславская	45286570000	55.780198	37.642653

		улица, дом 1, строение 1			

1. Общая характеристика программы

1.1 В обществе всё большее значение приобретает умение человека использовать компьютер не на пользовательском уровне, а на уровне начинающего программиста. В обязательном школьном курсе информатики программирование представлено на уровне, достаточном для прохождения экзамена, но не предполагает овладение практическими навыками применения языка. Следствием этого - формальное восприятие обучающимися основ современного программирования и неумение применять полученные знания на практике.

Программа «Программирование на С++» имеет техническую направленность, в её основу заложены принципы модульности и практической направленности, что обеспечит вариативность обучения. Содержание учебных модулей предполагает детальное изучение алгоритмизации, реализацию межпредметных связей, организацию проектной и исследовательской деятельности обучающихся.

Цель программы - формирование познавательной активности обучающихся в области функционального и объектно-ориентированного программирования, приобретение навыков работы с базовыми и сложными структурами языка в интегрированных средах разработки, получение навыков самостоятельного написания кода и разработки эффективных алгоритмов и программ.

1.2. Категории слушателей, на обучение которых рассчитана программа дополнительного образования (далее – программа): учащиеся 8-11 классов.

1.3. Нормативный срок освоения программы – 144 академических часа.

1.4. Форма обучения: очная с применением дистанционных образовательных технологий

Основной вид занятий – комбинированный, сочетающий в себе элементы теории и практики. Единицей учебного процесса является блок уроков (модуль). Каждый модуль охватывает отдельную информационную технологию или её часть. Внутри модулей разбивка по времени изучения производится педагогом самостоятельно, но с учётом рекомендованного календарно-тематического плана. Темп изучения отдельных разделов блока определяется субъективными и объективными факторами и необходимостью повторения.

2. Планируемые результаты обучения

Личностные результаты

– формирование ответственного отношения к учению, готовности и способности обучающихся к саморазвитию и самообразованию на основе мотивации к обучению и познанию, осознанному выбору и построению дальнейшей индивидуальной траектории образования на базе ориентировки в мире профессий и профессиональных предпочтений, с учётом устойчивых познавательных интересов;

– формирование целостного мировоззрения, соответствующего современному уровню развития науки и общественной практики, учитывающего социальное, культурное, языковое, духовное многообразие современного мира;

– формирование коммуникативной компетентности в общении и сотрудничестве со сверстниками, детьми старшего и младшего возраста, взрослыми в процессе образовательной, общественно полезной, учебно-исследовательской, творческой и других видов деятельности.

– развитие опыта участия в социально значимых проектах, повышение уровня самооценки благодаря реализованным проектам.

Метапредметные результаты

– умение самостоятельно определять цели своего обучения, ставить и формулировать для себя новые задачи в учёбе и познавательной деятельности, развивать мотивы и интересы своей познавательной деятельности;

– умение самостоятельно планировать пути достижения целей, в том числе альтернативные, осознанно выбирать наиболее эффективные способы решения учебных и познавательных задач;

– умение соотносить свои действия с планируемыми результатами, осуществлять контроль своей деятельности в процессе достижения результата, определять способы действий в рамках предложенных условий и требований, корректировать свои действия в соответствии с изменяющейся ситуацией;

– умение оценивать правильность выполнения учебной задачи, собственные возможности её решения;

– владение основами самоконтроля, самооценки, принятия решений и осуществления осознанного выбора в учебной и познавательной деятельности;

– умение определять понятия, создавать обобщения, устанавливать аналогии, классифицировать, самостоятельно выбирать основания и критерии для классификации, устанавливать причинно-следственные связи, строить логическое рассуждение, умозаключение (индуктивное, дедуктивное и по аналогии) и делать выводы;

– умение создавать, применять и преобразовывать знаки и символы, модели и схемы для решения учебных и познавательных задач;

– умение организовывать учебное сотрудничество и совместную деятельность с учителем и сверстниками; работать индивидуально и в группе: находить общее решение и разрешать конфликты на основе согласования позиций и учёта интересов; формулировать, аргументировать и отстаивать своё мнение;

– формирование и развитие компетентности в области использования информационно коммуникационных технологий.

Предметные результаты

– знание необходимой терминологии («данные», «команда», «алгоритм», «модель», «объект», «техническое задание»), смысла данных понятий и умение применять полученные знания на практике;

– знание об алгоритмических конструкциях и структурах данных;

– знание основных понятий и этапов проектной деятельности;

– умение соблюдать этикет программиста, не разрабатывать заведомо неработоспособный или приносящий вред программный код;

– умение соблюдать сетевой этикет, другие базовые нормы информационной этики и права при работе с компьютерными программами и в сети Интернет;

– умение составлять техническое задание на основе требований заказчика;

– умение разрабатывать программные решения, осуществлять их проектирование, разработку, тестирование, отладку и внедрение;

– развитие умений составить и записать алгоритм для конкретного исполнителя;

– навыки пошагового выполнения алгоритмов, умение осуществлять данные операции как вручную, так и с использованием программы отладки;

– навыки определения асимптотических оценок времени выполнения и затрат памяти для алгоритмов.

3. Структура программы
Учебный план

№	Тема (модуль)	Кол-во часов	В том числе		
			Аудит. занятия	Самост. работа	Промежуточная аттестация
	Модуль 1. Введение в C++	***36***	***18***	***14***	***4***
1.1	Введение. Достоинства и недостатки языка C++. Установка среды разработки. Типы переменных и арифметические операции	4	2	2	
1.2	Логические и битовые операции. Разветвления хода выполнения программы	4	2	2	
1.3	Приведение типов. Указатели и ссылки. Оператор запятая. Приоритет операций. Ввод и вывод данных	4	3	1	
1.4	Подробно про библиотеки. Циклы, 3 формы, их сравнение. Точки останова по условия. Вложенные циклы	4	2	2	
1.5	Время жизни переменных. Глобальные переменные. Динамическое выделение памяти. Массивы. Многомерные массивы	4	2	2	
1.6	Вспомогательные программы программиста, репозиторий. Функции и процедуры. Способы передачи аргументов в функцию	4	3	1	
1.7	Пространство имён. Рекурсия. Графы	4	2	2	
1.8	Оптимизация рекурсия. Поиск ответа перебором. Использование подхода динамического программирования. Работа с файлами	4	2	2	
	Промежуточная аттестация	4			4
	Модуль 2. STL и объектно-ориентированное программирование	***36***	***18***	***14***	***4***
2.1	Структуры, её поля и методы. Оперирование объектами. Разбиение проекта на объявление и реализацию	4	2	2	
2.2	Конструктор и деструктор. Константные методы. Композиция, декомпозиция и лист инициализации. Шаблонный тип данных. auto	4	3	1	
2.3	Перегрузка операторов. Функции, как параметры. std::vector, std::string, std::stack	4	2	2	
2.4	std::list, итераторы, поэлементный обход. Собственная реализация	4	2	2	
2.5	Пары и кортежи. std::queue, std::deque. Сортировка. Компаратор	4	2	2	
2.6	Библиотека <algorithm>. Бинарный поиск. Перестановка. Передача неограниченного числа параметров	4	2	2	
2.7	Бинарное дерево. Бор. std::set, std::multiset, std::map, std::bitset	4	3	1	
2.8	std::priority_queue. Случайные значения. Простые числа. std::unordered_map. Хеширование	4	2	2	
	Промежуточная аттестация	4			4

	Модуль 3. Проектирование приложений	***36***	***18***	***14***	***4***
3.1	Модификатор доступа. Наследование. Виртуальные методы. Абстрактный класс. Ассоциация объектов	5	3	2	
3.2	Применения навыков ООП. Дружественные функции. Модульное тестирование	4	3	1	
3.3	Работа с файловой системой и внешними ресурсами. Безопасные указатели. Исключения	5	3	2	
3.4	Многопоточность и многопроцессорность. Гонка ресурсов	5	3	2	
3.5	Применение параллельных вычислений. Работа с ядрами видеокарты. Матричные операции над варпами, блоками и сетками	5	3	2	
3.6	Работа в команде, распределение обязанностей. Управление временем	4	3	1	
	Индивидуальный промежуточный курсовой проект	4		4	
	Промежуточная аттестация	4			4
	Модуль 4. Оконные и специализированные приложения	***36***	***18***	***10***	***8***
4.1	Разработка оконных приложений	5	4	1	
4.2	Продолжение работы с оконными приложениями. Разделение приоритетов, распределение ролей в команде и составление плана работ	6	5	1	
4.3	Специализированное приложение	5	4	1	
4.4	Разработка приложения с учётом специфики	6	5	1	
	Командный промежуточный курсовой проект	2		2	
	Промежуточная аттестация	4			4
	Итоговый курсовой проект	8		4	4
Итого		***144***	***72***	***52***	***20***

4. Содержание программы

4.1. Содержание программы по модулям

Модуль 1. Введение в C++

Тема 1.1. Введение. Достоинства и недостатки языка C++. Установка среды разработки. Типы переменных и арифметические операции

Как и для каких целей появился язык C++, какая связь языка C++ и чистого СИ. Рассказ о том, в каких индустриях применяется язык C++ (можно упомянуть Unreal Engine и разработку компьютерных игр), а также о том, насколько эффективными получаются итоговые программные решения. Но, также стоит объяснить и о том, насколько сложно вести разработки на языке C++.

Установка среды разработки Visual Studio, создание первого консольного приложения. После этого объясняется, что из себя представляет наша программа и из каких основных частей состоит программный код.

Установка другой IDE, установка компилятора и отладчика, указание путей по умолчанию, связывание IDE со вспомогательными средствами.

Продемонстрировать возможности отладчика, пошаговое прохождение программы. использование точек остановки. Перечислить преимущества установленной среды разработки от онлайн аналогов (но при этом указав их, как альтернативы, возможные варианты: компилятор codeforces, GDB online Debugger и другие).

Перечислить возможные типы переменных (не забыть про unsigned), их различия, способ записи данных в битах, размеры. Продемонстрировать арифметические операции, объяснить их способ использования. Указать на то, что есть быстрые и медленные операции.

Тема 1.2. Логические и битовые операции. Разветвления хода выполнения программы

Написать несколько простых программ, на данный момент не обязательно демонстрировать консольный ввод\вывод. Рассмотреть логические и битовые операции. Продемонстрировать различие логических и битовых результатов (например, показав разницу результата при 1&2 и 1&&2).

Продемонстрировать способ использования условного оператора if. Дать объяснение оператору фигурные скобки (на данный момент можно условиться, что при помощи фигурных скобок группируются команды в единый выполняемый блок, не указывая на наличие локальной области видимости). Объяснить, что условие if приводится к типу данных bool. Показать возможность применения блока else. Продемонстрировать конструкцию множественного условия switch-case. Объяснить значение команды break и метки default.

Продемонстрировать использование команды goto, рассказать, в чём недостаток данной команды и про то, что любой код можно написать без её использования.

Тема 1.3. Приведение типов. Указатели и ссылки. Оператор запятая. Приоритет операций. Ввод и вывод данных

Рассказать про преобразование типов данных. Указать на то, что константам можно явно указывать тип данных при помощи соответствующего суффикса. Указать приоритет и направление выполнения операций. Продемонстрировать различие префиксного и суффиксного инкремента. Показать, как выбирается результирующий тип данных, а также возможные ситуации переполнения. Показать сложно составленные строки программного кода с несколькими действиями, задать вопросы на понимание, что получится в результате выполнения. Указать на отличие одинарной кавычки от двойной.

Показать тернарное условие и его основное применение (для вывода результирующего значения, а не для выполнения действий).

Рассказать про то, что у переменных имеются уникальные адреса. Абстрактно объяснить, как хранятся данные о переменных в памяти. Продемонстрировать работу с указателями. Показать операции получения

адреса и получения объекта по адресу. Показать возможность создание псевдонимов посредствам ссылок, а также то, как от указателей перейти к ссылкам. Показать преобразование типов данных посредством указателей, а также объяснить различие преобразования посредством оператора присваивания (то есть отличие от того, как преобразовывалось ранее).

Продемонстрировать использование оператора запятой. Показать, как выводится результирующее значения при применении оператора запятая.

Показать способ ввода и вывода данных в программу.

При помощи библиотеки iostream и консольного потока ввода\вывода cin\cout. Объяснить, что cin\cout - это сложно устроенные объекты, служащие буфером хранения данных из источника (в нашем случае это консоль) для ввода и параллельно для вывода.

При помощи scanf printf и библиотеки cstdio (или stdio.h в зависимости от компилятора). Рассказать, что scanf и printf - это функции, а что такое функции будет объясняться на следующих лекциях. Также отметить, что данные функции имеют неограниченное количество входных параметров.

Тема 1.4. Подробно про библиотеки. Циклы, 3 формы, их сравнение. Точки останова по условия. Вложенные циклы

Перечислить основные подключаемые библиотеки через #include и кратко дать их объяснение, чем они могут быть полезны.

Объяснить, что все команды, начинающиеся через # являются мета-командами. Показать, что такое макросы #define и условная компиляция #ifdef.

Реализовать циклы при помощи for, префиксного и постфиксного while. Научить оценивать асимптотику программы по времени и по памяти. Реализовать вложенные циклы, спросить у учащихся их асимптотики. Показать возможный вариант цикла с асимптотикой O(log(n)). Объяснить, почему в асимптотики у логарифма не ставится основание (из-за формулы перехода к другому основанию, показать её наличие). Сравнить асимптотики, дать оценку выполнения программы за 1 секунду (при каких входных данных успеет завершиться программа за 1 секунду с заданой асимптотикой).

Рассказать про оператор continue и способ написания циклов с выходом через условия.

Показать пример асимптотики O(sqrt(n)) (это может быть задача определения простоты перебором делителей до корня).

Тема 1.5. Время жизни переменных. Глобальные переменные. Динамическое выделение памяти. Массивы. Многомерные массивы

Рассказать про области видимости, задающиеся фигурные скобками, а также про время жизни статичных переменных (тех, что удаляются из памяти автоматически, когда закрывается их область видимости). Рассказать про глобальные переменные и их начальную инициализацию.

Рассказать про оператора new и динамическое выделение памяти, а также о том, как отличается время жизни динамических переменных от динамических.

Другие способы выделения памяти, например про функции из чистого СИ.

Рассказать, что такое оператор sizeof.

Рассказать, что такое утечки памяти и какой вред они могут нести. Научить отлавливать утечки памяти с помощью библиотеки CRT (<crtdbg.h>) Рассказать про команду delete.

Показать, как объявлять статичные массивы (на фиксированное количество элементов). Поговорить про то, как выбирать удачнее всего размер статичного массива, исходя из поставленной задачи. Показать, как объявлять многомерные массивы. Применить их в программе совместно с циклами и условиями.

Показать, как создавать динамический массив при помощи new и указателя. Рассказать про особенность использования char*, а также про то, какие ньюансы могут возникнуть при отладке (отладчик воспринимает указатель на char, как строку, ограниченную '\0').

Показать, насколько сложнее работать с динамическими многомерными массивами (потребуются циклы для объявления). Рассказать про преимущества использования динамических многомерных массивов во многих задачах (за счёт ступенчатого вида, а непрямоугольного, как в случае статических).

Продемонстрировать возможный вариант изменения размера динамического массива и переноса данных.

Показать, что оператор квадратные скобки есть смещение по адресу, хранимом в указателе.

Показать упрощённый вариант использования динамических массивов посредством std::vector.

Тема 1.6. Вспомогательные программы программиста, репозиторий. Функции и процедуры. Способы передачи аргументов в функцию

Показать возможный способ введения, планирования, проектирования и хранения проекта.

Сохранить в общий репозиторий весь ранее написанный код для свободного доступа к нему ученикам.

Объяснить, что такое функция, на примере простых действий (например, написав функцию суммирования или определения принадлежности символа заглавным). Указать на синтаксис описания функции. Рассказать про отличительную особенность функции main от остальных, а также о том, в чём смысл возвращаемого ею значения и аргументов. Напомнить про приведение типов при вызове функций. Показать, как применимы указатели при взаимодействии с функцией, чтобы изменять внешние для неё переменные.

Показать, как реализовать перегрузку функции. Объяснить, какая перегрузка применима и почему (та, что различима однозначно

компилятором). Указать на использование значений по умолчанию для параметров. Поговорить про константные переменные.

Показать, как передать оригинал в функцию посредством ссылки. Рассказать про копирование данных в переменные под аргументы, если не использовать ссылки. Показать, как сочетается константность и ссылка и зачем это используется на практике (чтобы не выделять новую память, но при этом, чтобы нельзя было изменить значение переменной).

Тема 1.7. Пространство имён. Рекурсия. Графы

Показать, как объявляется пространство. Объяснить, зачем оно используется. Рассказать про то, что основной функционал языка C++ находится в пространстве имён std.

Показать обозреватель объектов Visual Studio и то, как пространства имен упаковывают внутрь себя функции и глобальные переменные.

Объяснить на примерах, что такое рекурсия, где она встречается в повседневности (например, рекурсией оперирует файловая система, так в папке могут находиться другие папки). Показать, как через рекурсию можно переписать циклы.

Показать, как применить рекурсию для задач на перебор состояний и для других комбинаторных задач. Указать на то, что данные мы вводим слева-направо и снизу вниз, но при помощи рекурсии обработку данных можно изменить на сверху-вниз и слева-направо. Рассказать про то, что такое стек и что такое стек вызова функций. Изменить размер стека функций при помощи команды компилятору pragma comment(linker, "/STACK: "). Пройтись в отладчике по программе, использующей рекурсию.

Рассказать, что такое граф, как его можно хранить (3 формы представления) и как совершить по нему поиск в глубину при помощи рекурсии.

Совершить поиск пути в неявном графе (лабиринте) при помощи рекурсии.

Тема 1.8. Оптимизация рекурсия. Поиск ответа перебором. Использование подхода динамического программирования. Работа с файлами

Показать, как при помощи возвращаемого значения возможно остановить поиск при помощи рекурсии. Показать, как возможно восстановить ответ при помощи рекурсии.

Рассказать, что такое динамическое программирование. Показать примеры задач, например, нахождение чисел Фибоначчи или для задач подсчёта разбиений по указанному правилу. Совершить оптимизацию при помощи ленивой реализации динамического программирования.

Показать, как возможно считать и вывести данные в файл.

Показать перенаправление консольного потока на файл при помощи функции freopen. Показать защищенный вариант функции freopen_s, а также то, что такое проверка SDL Visual Studio, как её отключить и\или как добавить

отключение проверок на небезопасные функции (при помощи #pragma warning(disable : 4996)).

Рассказать про файловые потоки и библиотеку <fstream>. Рассказать про преимущества данного подхода работы данных в отличии от перенаправления потока.

Показать другие способы работы с файлами, например, через указатели на файл.

Рассказать про положительные и отрицательные стороны применения рекурсии.

Рассказать про синхронизацию старого ввода\вывода с новым. Отключить ей при помощи ios_base sync_with_stdio(false). Отключить синхронизацию ввода и вывода при помощи cin.tie(false), чтобы ускорить ввод и вывод данных.

Модуль 2. STL и объектно-ориентированное программирование

Тема 2.1. Структуры, её поля и методы. Оперирование объектами. Разбиение проекта на объявление и реализацию

Краткий рассказ о том, что из себя представляет объектно-ориентированное программирование, где используется оно, но также пояснить и то, где используется функциональное программирование. Показать синтаксис объявления структур, пока показывать только наличие полей. Указать на то, что теперь программа оперирует объектами, которые содержат совокупность полей. Написать несколько программ с использованием структур. Переписать предыдущие программы с использованием структур. Показать, как передавать структуры в функцию и что произойдёт при использовании глобальных структур, какова их начальная инициализация.

Показать что такое методы. Продемонстрировать их ассоциацию с объектом. Показать, что возможно динамически создавать структуры при помощи new. Показать, что такое оператор стрелочка. Разбить проект на файлы .cpp и .h, рассказать про пользу отделения описания от реализации.

Тема 2.2. Конструктор и деструктор. Константные методы. Композиция, декомпозиция и лист инициализации. Шаблонный тип данных. Auto

Рассказать, что такое конструктор, как его объявить. Рассказать про конструктор по умолчанию и его перекрытие собственным конструктором. Рассказать про то, что для функций (но не для методов) без параметров не требуется указывать круглые скобки. Продемонстрировать вызов конструктора при помощи отладки.

Деструктор, показать его объявление. Продемонстрировать вызов деструктора при помощи отладки, а также для динамических объектов (при помощи new и delete). Показать наиболее частые случаи использования деструктора (для избежания утечек памяти).

Рассказать про объявление и назначение константных методов.

Продемонстрировать композицию структуры при помощи списка инициализации. Продемонстрировать разбиение структуры на переменные

при помощи декомпозиции (стандарт CPP17). Показать при этом, что создаётся копия, а для обращения к оригиналу нужно использовать ссылку.

Продемонстрировать, как использовать шаблонный тип данных для структуры и для функций. Показать, какие нюансы возникают при использовании шаблонов (например, несовпадения типа данных аргументов и невозможность их выбора, как в std::min при разных аргументах). Показать, что такое тип auto и как он может быть использован в описании функции (для параметров и для результирующего значения).

Указать на то, что шаблонные структуры не могут иметь реализацию в другом файле, где нет объявления.

Рассказать про изменения, которые затронули auto в различных стандартах языка.

Рассказать более подробно про список инициализации, его блокировку приведения типов и использование при описании конструктора, а также, какие плюсы возникают при его использовании в конструкторе.

Показать, как возможно отделить объявление от реализации для шаблонных структур. Показать, как возможно использовать intelliSense для указания явных типов у шаблона.

Тема 2.3. Перегрузка операторов. Функции, как параметры. std::vector, std::string, std::stack

Показать синтаксис перегрузки операторов. Показать набор возможных перегрузок. Продемонстрировать для своих структур.

Рассказать, что такое void*. Показать, как возможно передать фунцию, как параметр. При помощи библиотеки <functional> и структуры std::function. Продемонстрировать примерами. При помощи стандартного синтаксиса и при помощи void*. Показать, что vector является шаблонной структурой с перегруженными операторами. Реализовать собственный vector (с конструктором и деструктором). Показать методы vector, а также string и stack. Написать перегрузку оператора >> для считывания vector при помощи стандартных потоков. Показать, как возможно считать данные построчно, посимвольно и до конца файла. Показать, как string преобразовать в поток std::stringstream и записать из него данные в переменные. показать возможности библиотеки <iomanip> и вывода вещественного числа с указанной точностью (а также как точность указывается в scanf).

Тема 2.4. std::list, итераторы, поэлементный обход. Собственная реализация

Объяснить различие std::list от std::vector, завести таблицу с указанием скорости выполнения операций для разных структур данных. Показать, как list хранится в памяти.

Итераторы, объяснить их назначение. Рассказать про виды итераторов и про то, как ими пользоваться в программе.

Показать вставку по итератору, и что происходит при удалении по итератору.

Обратные итераторы (и как они конвертируются друг в друга).

Показать поэлементный обход по структурам. Показать различие копирования данных в поэлементном обходе, от обращения по ссылки. Продемонстрировать декомпозицию в поэлементном обходе. Осуществить собственную реализацию двусвязного (или односвязного) списка.

Реализовать свои итераторы. Показать функции std::next, std::prev, std::advance и std::distance.

Тема 2.5. Пары и кортежи. std::queue, std::deque. Сортировка. Компаратор

Показать возможность применения нескольких шаблонных типов, а также как оперировать с std::pair. Показать, как использовать std::tuple и функцию std::get. Рассказать про структуры данных std::queue и std::deque. Оценить время выполнения операций.

Рассказать, как хранится в памяти std::deque. Сделать собственную реализацию.

Способы сортировки, начиная с медленных (O(n^2)). Рассказать про сортировку подсчётом.

Рассказать про быстрые сортировки (O(n*log(n)), например про сортировку слиянием.

Реализовать быструю сортировку (разделяй-и-властвуй, она же Хоара).

Возможные оптимизации сортировки, например с использованием троичного разбиения и выбора случайного опорного элемента. Для простых сортировок можно отметить, когда возможно досрочное прекращение.

Показать встроенные в STL сортировки (std::sort и другие, зависит от компилятора). Показать, что такое компаратор-функция и что такое компаратор-структура. Применить компаратор для std::sort (упомянуть про std::less и std::greater).

Тема 2.6. Библиотека <algorithm>. Бинарный поиск. Перестановка. Передача неограниченного числа параметров

Показать реализованные функции в библиотеки <algorithm>.

Объяснить, что такое бинарный поиск, как он реализуется. Показать левосторонний и правосторонний бинарный поиск, показать их отличие (когда есть одинаковые элементы выбираются разные крайние позиции). Показать бинарный поиск по ответу (то есть подбор варианта, но с сокращением пространства поиска в два раза).

Показать, как использовать std::lower_bound и std::upper_bound, их отличия друг от друга, когда функции применимы.

Рассказать, что такое тернарный (троичный) поиск. Возможно, привести пример рычажных весов и поиска фальшивой монеты.

Рассказать, что такое перестановка. Воспользоваться функцией next_permutation для поиска следующей перестановки.

Рассказать, что такое constexpr и зачем он применяется.

Реализовать собственную функцию генерации перестановок (или получения соседней).

Передача неограниченного количества аргументов в функцию, например, при помощи va_list.

Метапрограммирование и предподсчёт на этапе компиляции. Передача неограниченного количества шаблонных типов.

Тема 2.7. Бинарное дерево. Бор. std::set, std::multiset, std::map, std::bitset

Построить бинарное дерево поиска (следуя правилу, что меньший стоит добавлять слева, больший справа, с дубликатами как обращаться решает преподаватель). Показать, как его можно использовать и какие проблемы возникают при добавлении и удалении элементов. Показать дерево Бор, решить некоторые простые задачи с его использованием (например, хранение телефонной книги).

Рассказать про структуру данных std::set, её методы, указать в общей таблице асимптотику выполнения операций.

Рассказать, что такое красно-чёрное (или АВЛ) дерево, его свойства, примерно указать, как реализуются команды добавления и удаления в дерево (рассказать про наличие поворотов).

Передать компаратор в std::set. Продемонстрировать решение задач при помощи std::set. Рассказать про наличие std::multiset, его сходство и то, как альтернативно можно было реализовать multiset через обычный set (с использованием pair). Рассказать про итераторы set и про запрет на изменение значений напрямую.

Рассказать про std::map, сравнить его с set, показать добавочные операции (перегруженные квадратные скобки) и как его применяют. Решить задачи с его использованием. Сделать поэлементный обход с возможностью изменения значений. Добавить компаратор.

Указать ситуации, когда set и map не удастся использовать (есть не сравниваемые оператором меньше собственные структуры). Использовать set и map для ранее написанных собственных структур.

Указать в общей таблице (где сравниваются структуры данных STL) асимптотику выполнения разобранных структур данных.

Рассказать про bitset, то, как можно использовать переменные для тех же задач, как инвертировать определённые биты.

Тема 2.8. std::priority_queue. Случайные значения. Простые числа. std::unordered_map. Хеширование

Рассказать про структуру данных std::priority_queue, ей методы, когда используется. Также рассказать про функцию make_heap.

Сравнить скорости выполнения set и priority_queue для одной и той же задачи (например, для алгоритма Дейкстры). Для измерения расходов памяти и требуемого времени можно воспользоваться средством диагностики Visual Studio.

Рассказать, что такое структура данных куча, ей свойства, как она устроена. Возможно упомянуть в том числе про Декартово дерево.

Рассказать про функцию rand и функцию srand, а также про возможность получения текущего времени при помощи функции time. Рассказать про библиотеку <random> и про используемые в ней генераторы (например, для получения long long).

Рассказать, что представляют из себя простые числа. Предложить алгоритм нахождения простых чисел, а также алгоритм проверки за простоту (желательно с асимптотикой O(sqrt(n)).

Рассказать про Китайскую теорему об остатках и её конструктивное применение.

Рассказать про std::unordered_map, её методы и для каких данных применима (только для тех, что имеют алгоритм хеширования). Рассказать про асимптотику и про скрытую константу, а также про случаи, когда unordered_map эффективнее map. Рассказать про коллизии и про опасность увеличения асимптотики с O(log(n)) до O(n).

Рассказать про алгоритм хеширования и про то, как устроены хеш-таблицы. Сделать собственную реализацию одним из способов.

Полиномиальный хеш и когда он применим, для каких задач.

Модуль 3. Проектирование приложений

Тема 3.1. Модификатор доступа. Наследование. Виртуальные методы. Абстрактный класс. Ассоциация объектов

Построить концепцию ООП с учащимися. Рассказать про модификаторы доступа, в чем отличие protected от private. Рассказать, в чём отличие класса от структур (в модификаторе доступа, но можно упомянуть о различиях, которые были в чистом СИ). Продемонстрировать наследование, рассказать, когда оно применяется. Продемонстрировать мультинаследование и рассказать, какой от него вред, какие проблемы проектирования могут возникнуть при мультинаследовании. Рассказать про виртуальные методы, в чём их назначение. Рассказать про Абстрактные классы и чисто виртуальные методы. Показать различие между композицией и агрегацией при проектировании проекта. Показать UML диаграммы. Возможно, продемонстрировать построение UML диаграмм при помощи Microsoft Visio. Рассказать про известные ООП проекты и их архитектуры, UML диаграммы.

Тема 3.2. Применения навыков ООП. Дружественные функции. Модульное тестирование

Применить навыки ООП для реализации собственной структуры, которой нет в STL. Продемонстрировать концепцию getter и setter методов. Применить навыки ООП для реализации структуры "Длинные числа". Перегрузить операции. Используем обратный код для выполнения вычитания чисел. Используем дружественные функции для того, чтобы обратиться к приватным полям длинного числа. Реализуем операцию деления и нахождения длинного корня при помощи бинарного поиска. Реализовать умножение через

Быстрое преобразование Фурье. Реализовать другую структуру. Возможно, это будет дерево отрезков, возможно, это будет та структура, которая понадобится для итогового проекта (например, структура верстки для написания веб сайта). Применить описание дружественных функций. Показать, как осуществить ассемблерную вставку. Рассмотреть возможную реализацию при помощи сторонней библиотеки (возможно библиотеки boost). Показать, как применять модульное тестирование.

Тема 3.3. Работа с файловой системой и внешними ресурсами. Безопасные указатели. Исключения

Воспользоваться библиотекой <filesystem> для решения прикладных задач. Осуществить работу с несколькими файлами. Это могут быть ресурсы, которые будут использованы для итогового проекта. В случае компьютерной игры это могут быть изображения и другие медиа ресурсы, в случае веб сайта это могут быть html страницы. Для научной таблицы это могут быть таблицы в формате .csv. Применить умные указатели (библиотека <memory>). Показать отличия между std::unique_ptr , std::shared_ptr , std::weak_ptr. Продемонстрировать отсутствие утечек памяти. Рассказать про исключения, библиотеку <stdexcept> конструкцию try catch. Обратить внимание на то, что процесс выполнения не прерывается командой throw, если нет внешнего try. Рассказать про модификатор функции noexcept. Рассказать про библиотеку <typeinfo> и определения типа передаваемого параметра (может оказаться полезным в случае шаблонной функции или использования типа auto).

Тема 3.4. Многопоточность и многопроцессорность. Гонка ресурсов

Создание многопоточных приложений. Использовать библиотеку <thread>. Импортировать библиотеку boost. Рассказать про гонку ресурсов, мьютексы и семафоры. Показать их схожесть с умными указателями. Рассказать, в чём отличие многопотоковых от многопроцессорных приложений. Создание многопроцессорного приложения с использованием любой удобной библиотеки (рекомендуется OpenMP). Показать передачу данных в многопроцессорных приложениях при помощи пересылки сообщений.

Тема 3.5. Применение параллельных вычислений. Работа с ядрами видеокарты. Матричные операции над варпами, блоками и сетками

Продолжение объяснения материала параллельного программирования. Демонстрация возможностей многопроцессорных программ. Составление программ, обрабатывающих запросы параллельно. Продолжения изучения материала предыдущего занятия. Рассмотрение возможности выполнения операций с использованием видеокарты. Использование библиотеки для работы с CUDA ядрами. Объяснение нюансов использования ресурсов видеокарты, специфичности обработки данных и применяемых операций (объяснить, что выполняются операции над многомерными матрицами

большой размерности, под каждый варп выделяется отдельный поток, поэтому операции над каждым элементом происходят синхронно).

Тема 3.6. Работа в команде, распределение обязанностей. Управление временем

Время проектирования и жизнь проекта, какие этапы развития проходит проект. Занятие строится на опыте преподавателя. Рассказать про средства совместной разработки, про общий репозиторий и его разветвление. Рассмотреть механизм слияния ветвей в общем репозитории. Основные обязанности в команде для реализации проекта заданной итоговой работой направленности (если это реализация компьютерной игры, то это дизайнер, аниматор, проектировщик внутренней логики, интерфейс-дизайнер и другие роли, если это разработка, то это Frontend и Backend разработчик).

Модуль 4. Оконные и специализированные приложения

Тема 4.1. Разработка оконных приложений

Подготовка к созданию оконных приложений. Установка подходящей среды разработки и дополнительных библиотек. Установка QT Creator. Настройка компилятора и отладчика. Настройка библиотеки MFC. Использование библиотеки SFML. Использование другой библиотеки, на усмотрение преподавателя. Создание первого оконного проекта. Проверка успешности сборки. Рассказ об аналогии создания оконных приложений на других языках программирования, например на C#. Объяснение разделения интерфейса пользователя от логики проекта. Выполнение ранее поставленных задач посредством оконного приложения (преобразование ранее написанных консольных приложений в оконные).

Тема 4.2. Продолжение работы с оконными приложениями. Разделение приоритетов, распределение ролей в команде и составление плана работ

Разработка оконных приложений. Рассмотрение часто встречаемых ошибок. Управление оконными свойствами. Рассмотрение набора оконных элементов. Разбор паттернов разработки: события, делегаты, фабричный метод, фасад. Определить цель и задачи последующего курсового проекта, определить то, что должна выполнять оконная программа по итогу выполнения. Распределение учащихся по командам, распределение их обязанностей. Установление сроков выполнения (желательно с указанием 3 оценок временных интервалов по схеме PERT, но возможно и другие способы, известные преподавателю).

Тема 4.3. Специализированное приложение

Определить новую среду разработки или библиотеку для реализации итогового курсового проекта. В зависимости от итогового курсового проекта и специализации преподавателя. Это может быть среда или библиотека для разработки графических приложений, таких как компьютерных игр или построителей графиков. Возможно это будет среда для разработки веб-приложения, мобильного приложения, высокопроизводительного приложения для научных вычислений или разработки искусственного интеллекта. Как вариант, средой разработки может являться Unreal Engine или возможно потребуется библиотека SFML. Возможен вариант подключения библиотеки для работы с базой данных. Показать учащимся, как взаимодействовать с новой средой разработки и как разрабатывать в ней приложение.

Тема 4.4. Разработка приложения с учётом специфики

В зависимости от выбранной специфики: показать возможности встроенных функций. Если это новая среда разработки, показать встроенный функционал, вывести наиболее часто используемые в работе окна обозревателя. Показать, как найти и пользоваться документацией. Разобрать возможные гибкие решения, для упрощения дальнейшей модификации проектов. Распределить учащихся по командам или дать индивидуальные задания (возможен смешанный вариант). Предложить темы итогового курсового проекта.

4.2. Примеры заданий для организации самостоятельной работы слушателей

Модуль 1

Реализовать консольное приложение, выполняющее арифметические, битовые и логические операций над данными, введёнными пользователем при помощи консоли и\или при помощи файла. В качестве входных данных передаётся строка длиной не превышающая 1000 символов, состоящая из цифр, знаков операций (соответствующие синтаксису C++) и круглых скобок. Операцию деления требуется производить в поле целых чисел (целочисленное деление).

Рекомендации по выполнению: перезаписать входные данные в соответствии обратной польской записи, использовать рекурсию для раскрытия круглых скобок.

Модуль 2

Спроектировать UML модель проектного решения в выбранной области проблематики. Реализовать консольное приложение в объектно-ориентированном стиле программирования в соответствии с построенной UML моделью.

При проектировании требуется использовать гибкие решения (зарекомендовавшие себя паттерны). Проверка гибкости будет осуществлять сменой требований условий заказчика и количественной проверкой изменений в преобразованной UML модели.

Рекомендации по выполнению: для успешного выполнения задания требуется рассмотреть различные сценарии изменения условий заказчика и спроектировать своё программное решение таким образом, чтобы оно было масштабируемым, независимым от интерфейса и модульным.

Модуль 3

Скоординировать командную работу над проектом, разбить итоговую задачу на составные части. Распределить обязанности в команде и сроки выполнения каждой части проекта. Спроектировать UML модель каждой части проектного решения в выбранной области проблематики. Составить план совмещения частей проекта воедино. Реализовать консольное приложение в объектно-ориентированном стиле программирования в соответствии с выбранным разбиением на модули и построенной UML моделью. Совместить все модули воедино, совершить модульное и системное тестирование. Представить рабочий проект.

Рекомендации по выполнению: для успешного выполнения задания требуется сформировать навыки командного взаимодействия, а также навыки организации времени.

Модуль 4

Спроектировать UML модель проектного решения в выбранной области проблематики. Реализовать графическое приложение в объектно-ориентированном стиле программирования в соответствии с построенной UML моделью. При проектировании требуется использовать гибкие решения (зарекомендовавшие себя паттерны). Осуществить разделение данных приложения от исполняемого кода. Совершить Frontend и Backend тестирование. Указать перспективы развития приложения.

Рекомендации по выполнению: для успешного выполнения задания требуется критически оценить свои возможности и ресурсы для выбора конечной цели и итогового варианта приложения.

4.3. Список рекомендуемой литературы

1. Поляков К. Ю., Еремин Е. А. Информатика. Углублённый уровень. Учебник для 10 класса в 2 частях. М.: БИНОМ. Лаборатория знаний, 2014.

2. Информатика и ИКТ. Задачник-практикум в 2 частях. Под ред. И. Г. Семакина и Е. К. Хеннера. М.: БИНОМ. Лаборатория знаний, 2014.

3. Лааксонен А. Олимпиадное программирование: ДМК Пресс, 2022.

4. Мартин Р. Идеальный программист. Как стать профессионалом разработки ПО: СПб. Питер, 2011.

5. Гамма Э. и др. Приемы объектно-ориентированного проектирования. Паттерны проектирования: Учебное пособие, 2007.

6. Халим С., Халим Ф. Спортивное программирование: ДМК Пресс, 2022

7. Кнут Д. Э. Искусство программирования: Издательский дом Вильямс, 2000.

Электронные ресурсы:

1. Мирзаянов М. Платформа для соревнований по программированию Codeforces. 2013. [Электронный ресурс] URL: https://codeforces.com/ (дата обращения: 10.08.2022).

2. Престон-Вернер Т. И др. Веб-сервис для хостинга IT-проектов и их совместной разработки GitHub. 2008. [Электронный ресурс] URL: https://github.com/ (дата обращения: 10.08.2022).

3. Спольски Д. Визуальный веб-инструмент управления проектами, рабочими процессами и заданиям Trello. 2011. [Электронный ресурс] URL: https://trello.com/ (дата обращения: 10.08.2022).

5. Материально-технические условия реализации программы

Требования к помещению:

– помещение для занятий, отвечающие требованиям СанПин для учреждений дополнительного образования;

– качественное освещение;

– столы, стулья по количеству обучающихся и 1 рабочим местом для педагога.

Информационное обеспечение:

– операционная система (желательно Windows);

– браузер: Yandex Browser, Chrome, Chrome Mobile, Firefox, Opera, Safari, Mobile Safari, Edge;

– среда разработки программного обеспечения: Microsoft Visual Studio, Code::blocks, CLion, Visual Studio Code.

Требования к оборудованию

Таблица 4

Наименование специализированных аудиторий, кабинетов, лабораторий	Вид занятий	Наименование оборудования, программного обеспечения
Аудитория с доступом в Интернет	Аудиторные занятия	Компьютер, среда разработки программного обеспечения, проектор, видеокамера, доступ в Интернет
Аудитория с доступом в Интернет	Самостоятельная работа	Персональные компьютеры/ноутбуки, среда разработки программного обеспечения, доступ в Интернет

6. Оценка качества освоения программ

Оценка качества освоения программы осуществляется в процессе промежуточной аттестации.

Формы и методы промежуточного контроля представлены в таблице 5.

Таблица 5

Наименование модулей	Основные показатели оценки	Формы и методы контроля и оценки
Модуль 1. Введение в C++	Установленное количество выполненных заданий	Самостоятельное выполнение заданий
Модуль 2. STL и объектно-ориентированное программирование	Установленное количество выполненных заданий	Самостоятельное выполнение заданий
Модуль 3. Проектирование приложений	Установленное количество выполненных заданий	Самостоятельное выполнение заданий
Модуль 4. Оконные и специализированные приложения	Установленное количество выполненных заданий	Самостоятельное выполнение заданий

7. Составители программы

Халтурин Евгений Александрович, ассистент кафедры ИС ИКИТ СФУ

Малеев Алексей Викторович, директор Центра развития ИТ-образования

Рухович Филипп Дмитриевич, к. физ.-мат. н., доцент кафедры АТП ФПМИ МФТИ

Мартемьянов Роман Юрьевич, заместитель директора Центра развития ИТ-образования

Сырцова Елена Леонидовна, PhD, к. пед. н., доцент, руководитель проектов Центра развития ИТ-образования

Токмакова Ольга Викторовна, PhD, к. пед. н., доцент, специалист по учебно-методической работе Центра развития ИТ-образования

Зуева Наталья Александровна, менеджмер проектов Центра развития ИТ-образования

Согласовано
Ведущий специалист отдела сопровождения образовательных программ

_____ Ж.И. Зубцова

Согласовано
Директор ЦРИТО

_____ А.В. Малеев