

**Федеральное государственное автономное образовательное  
учреждение высшего образования  
«Московский физико-технический институт  
(национальный исследовательский университет)»**

**УТВЕРЖДЕНО**

**Директор физтех-школы  
прикладной математики и  
информатики**

**А.М. Райгородский**

	<b>Рабочая программа дисциплины (модуля)</b>
<b>по дисциплине:</b>	Программирование на языке C++. Продвинутый поток
<b>по направлению:</b>	Прикладная математика и информатика
<b>профиль подготовки:</b>	Анализ данных в экономике
	Физтех-школа Прикладной Математики и Информатики кафедра алгоритмов и технологий программирования
<b>курс:</b>	1
<b>квалификация:</b>	бакалавр

Семестр, формы промежуточной аттестации: 1 (осенний) - Дифференцированный зачет

Аудиторных часов: 30 всего, в том числе:

лекции: 15 час.

семинары: 15 час.

лабораторные занятия: 0 час.

Самостоятельная работа: 60 час.

Всего часов: 90, всего зач. ед.: 2

Количество контрольных работ, заданий: 2

Программу составил: И.С. Мещерин, ассистент

Программа обсуждена на заседании кафедры алгоритмов и технологий программирования 12.02.2024

## Аннотация

Курс содержит в себе принципы объектно-ориентированного подхода, а так же объектно-ориентированный анализ и проектирование. Раскрывает и описывает основные подходы в объектно-ориентированном программировании. Рассматривает обобщенные классы и методы. Строго типизированные источники и управление взаимодействием объектов.

### 1. Цели и задачи

#### Цель дисциплины

- сформировать представление о разнообразных вычислительных задачах в теории графов и об асимптотических сложностях их решений;
- дать теоретические и практические знания об алгоритмах и структурах данных теории графов с доказательством корректности их работы, о методах оценки сложности алгоритмов.

#### Задачи дисциплины

- научить формулировать задачи в терминах изученных теорий, выбирать подходящий алгоритм для поставленной задачи;
- научить разрабатывать комбинации алгоритмов для решения поставленных задач, оценивать сложности алгоритмов, их модификаций и комбинаций, в том числе с помощью амортизационного анализа, выбирать подходящие структуры данных для поставленных задач, реализовывать алгоритмы в обобщенной форме на языке программирования C++.

### 2. Перечень формируемых компетенций

Освоение дисциплины направлено на формирование следующих компетенций:

Код и наименование компетенции	Индикаторы достижения компетенции
ОПК-1 Способен применять фундаментальные знания, полученные в области физико-математических и (или) естественных наук и использовать их в профессиональной деятельности	ОПК-1.2 Способен строить математические модели, производить количественные расчеты и оценки
ОПК-2 Способен использовать современные информационные технологии и программные средства при решении задач профессиональной деятельности, соблюдая требования информационной безопасности	ОПК-2.2 Знает и умеет применять численные математические методы и прикладное программное обеспечение для решения научных задач в профессиональной области

### 3. Перечень планируемых результатов обучения по дисциплине (модулю)

В результате освоения дисциплины обучающиеся должны

знать:

- алгоритмы на графах и структуры данных, связанные с ними;
- оценки сложности стандартных алгоритмов;
- стандартные алгоритмы на графах и используемые структуры данных, подходы к модификации классических алгоритмов;
- разнообразные классические задачи в теории графов и асимптотические сложности их решений.

уметь:

- формулировать задачи в терминах изученных теорий, выбирать подходящий алгоритм для поставленной задачи;
- разрабатывать комбинации алгоритмов для решения поставленной задачи;
- оценивать сложности алгоритмов, их модификаций и комбинаций, в том числе с помощью амортизационного анализа;
- выбирать подходящие структуры данных для конкретной задачи;
- реализовывать алгоритм в обобщенной форме на языке программирования c++;
- реализовывать стандартные алгоритмы на графах и структуры данных на языке программирования C++.

владеть:

- методами декомпозиции задач в области информационных технологий и построения единого решения с использованием изученных алгоритмов;
- методами оценки сложности алгоритмов, их модификаций и комбинаций.

#### 4. Содержание дисциплины (модуля), структурированное по темам (разделам) с указанием отведенного на них количества академических часов и видов учебных занятий

##### 4.1. Разделы дисциплины (модуля) и трудоемкости по видам учебных занятий

№	Тема (раздел) дисциплины	Трудоемкость по видам учебных занятий, включая самостоятельную работу, час.			
		Лекции	Семинары	Лаборат. работы	Самост. работа
1	Введение в язык	1	1		7
2	Модификаторы типов	2	2		8
3	Введение в ООП	2	2		7
4	Перегрузка операторов	2	2		8
5	Наследование (inheritance)	2	2		7
6	Шаблоны (templates)	2	2		8
7	Исключения (exceptions)	2	2		7
8	Аллокатеры (allocators)	2	2		8
Итого часов		15	15		60
Подготовка к экзамену		0 час.			
Общая трудоёмкость		90 час., 2 зач.ед.			

##### 4.2. Содержание дисциплины (модуля), структурированное по темам (разделам)

###### Семестр: 1 (Осенний)

###### 1. Введение в язык

Общие слова: место языка C++ среди современных языков, актуальные версии этого языка, ключевые люди, связанные с этим языком, официальный стандарт языка

Структура программы, функция main, понятие области видимости (scope). Ключевые слова. Ввод-вывод (cin, cout).

Объявления (declarations). Идентификаторы. Фундаментальные типы (int, long, long long, float, double, long double, char, bool, модификаторы signed и unsigned). Размеры этих типов, основные операции над ними, неявные преобразования типов между собой. Литералы, литеральные суффиксы для основных типов. Объявления функций, разница между объявлением и определением, one definition rule.

Выражения (expressions). Операторы. Арифметические операторы. Побитовые операторы. Логические операторы, особенности их работы. Оператор присваивания и операторы составного присваивания, особенности его работы. Понятие lvalue и rvalue в C++03. Инкремент и декремент, отличие префиксной версии от постфиксной. Операторы сравнения. Тернарный оператор. Оператор “запятая”. Оператор sizeof.

Инструкции (statements). Конструкции if...else, for, while, do...while, switch, их синтаксис, правила работы. Инструкции break, continue, return, их действие. Инструкция goto и метки.

Понятия ошибки компиляции, ошибки времени выполнения (runtime error), неопределенного поведения (undefined behaviour), отличия между ними, примеры. Виды ошибок компиляции: лексические, синтаксические, семантические. Понятие segmentation fault и stack overflow.

###### 2. Модификаторы типов

Указатели, операции над ними. Операция взятия адреса. Автоматическая память (стек). Массивы, операция [] (квадратные скобки), ее принцип работы. Указатель на void и его особенности.

Функции. Перегрузка функций, правила разрешения перегрузки (общая схема, без деталей). Функции с аргументами по умолчанию. Функции с неуказанным количеством аргументов. Указатели на функции, операции над ними, их особенности.

Динамическая память. Операторы new и new[], их использование (в стандартной форме). Операторы delete и delete[], их использование (в стандартной форме). Проблема утечек памяти. Проблема двойного удаления.

Передача аргументов по значению и по указателю, первая версия функции swap. Дилемма с присваиванием (создавать новое название или копию?). Идея ссылок (references). Отличия ссылок от указателей, правила работы со ссылками, вторая версия функции swap. Проблема, связанная со ссылкой на локальную переменную (“битые ссылки”).

Идея констант, ключевое слово const. Понятие константных и неконстантных операций, особенности работы с константами. Константные и неконстантные ссылки. Константные указатели и указатели на константу. Разрешенные и запрещенные присваивания между всеми вышеупомянутыми типами.

Виды приведений типов: static\_cast, reinterpret\_cast, const\_cast и C-style cast, их особенности, примеры применения и примеры, когда они не работают.

### 3. Введение в ООП

Идея ООП. Понятия класса и структуры, членов класса. Поля и методы, понятие инкапсуляции. Модификаторы доступа.

Конструкторы и деструкторы. Конструктор по умолчанию. Перегрузка конструкторов. Конструктор копирования, его сигнатура и схема реализации. Пример, когда необходим нетривиальный конструктор копирования и оператор присваивания. Правила генерации компилятором конструкторов. Ключевые слова default и delete в контексте определения функций-членов.

Операторы “точка” и “стрелочка”. Ключевое слово this и пример использования.

Оператор присваивания, его сигнатура и схема реализации. “Правило трех”.

Проблема с инициализацией констант и ссылок. Решение: списки инициализации в конструкторах.

Ключевое слово explicit. Пример с конструктором String(int n).

Константные и неконстантные методы, примеры.

Ключевое слово mutable, пример применения.

Понятие дружественных функций и классов, ключевое слово friend.

Проблема вызова конструкторов из других конструкторов. Решение: делегирующие конструкторы.

Статические поля и методы, пример. Локальные статические переменные.

Указатели на члены и указатели на методы. Синтаксис объявления, пример использования.

Операторы “точка со звездочкой” и “стрелочка со звездочкой”.

### 4. Перегрузка операторов

Общая идея перегрузки операторов. Перегрузка арифметических операторов на примере класса BigInteger: бинарные операторы, составные присваивания с ними, правильное выражение одного через другое. Проблема с корректностью выражений вида “x+y=5;”. Проблема в случае левого операнда - не объекта класса (выражения вида “5+x”). Перегрузка операторов << и >> на примере потокового ввода-вывода.

Перегрузка операторов сравнения, правильное выражение одних сравнений через другие.

Перегрузка инкремента и декремента (префиксного и постфиксного).

Перегрузка оператора [] (квадратные скобки). Правильное соблюдение константности при перегрузке оператора [].

Перегрузка оператора “круглые скобки”. Понятие функтора и функционального класса, компаратора. Пример использования в стандартных алгоритмах.

Особенности перегрузки операторов “логическое И”, “логическое ИЛИ” и “запятая”.

Особенности перегрузки операторов “унарная звездочка”, “унарный амперсанд” и “стрелочка”.

Перегрузка операторов приведения типа. Еще одно применение ключевого слова `explicit`.

## 5. Наследование (inheritance)

Объявление наследования. Модификатор доступа `protected`. Разница между приватным, публичным и защищенным наследованием. Разница между наследованием классов и структур. Поиск имен при наследовании. Соккрытие имен наследником. Явный вызов методов родителя у наследника. Использование `::` и `using`. Проблемы с видимостью названий родителей и их полей у потомков в случае двухуровневого наследования, где первый уровень - приватное наследование. Правила действия слова `friend` в этих случаях.

Порядок вызова конструкторов и деструкторов при наследовании. Проблема с инициализацией родителей при определении конструктора наследника, вновь применение списков инициализации. Правила размещения объектов классов-наследников в памяти.

Множественное наследование, неоднозначности при нем, проблема ромбовидного наследования. Примеры разрешения неоднозначности с помощью приведений типов и оператора `::`, комбинации всего этого с приватным наследованием, сдвиги указателей.

Виртуальное наследование. Особенности комбинации виртуального и неvirtуального наследования.

Приведение типов между родителем и наследником: срезка при копировании, приведение указателей, приведение ссылок. Особенности `static_cast`, `reinterpret_cast` между родителями и наследниками (а также указателями или ссылками на них). Оператор `dynamic_cast`, его отличие от `static_cast`.

Виртуальные функции, их общая идея и отличие от неvirtуальных. Особенности размещения в памяти классов с виртуальными функциями, понятие полиморфизма. Полиморфные классы. Понятие о таблице виртуальных функций.

Виртуальный деструктор и его предназначение.

Абстрактные классы и “чисто виртуальные” (pure virtual) функции, их особенности. Чисто виртуальный деструктор. Ошибка “pure virtual function call” и ее возникновение.

Ключевые слова `override` и `final` при наследовании, их предназначение.

Механизм RTTI. Оператор `typeid` и динамическое определение типа объекта. Класс `std::type_info`.

Проблема с вызовом виртуальных функций в конструкторах. Проблема с аргументами по умолчанию в виртуальных функциях.

Empty base optimization, примеры.

## 6. Шаблоны (templates)

Мотивировка и общая идея шаблонов. Шаблоны классов, шаблоны функций, синтаксис объявления, примеры использования, связь шаблонов и полиморфизма, статический полиморфизм.

Специализации шаблонов, принцип “частное предпочтительнее общего” применительно к шаблонам. Частичные и полные специализации. Принцип “лучше точное соответствие, чем приведение типа”. Разница между специализацией и перегрузкой для шаблонных функций. Правила выбора компилятором кандидатов на специализацию и на перегрузку.

Ключевое слово `typedef`, его предназначение. Шаблонные `typedef`’ы, использование слова `using`.

Проблема с обращением к `typedef`’ам внутри шаблонных классов. Применение ключевого слова `typename` для решения этой проблемы.

Примеры реализации простейших `type_traits` с помощью шаблонных структур и `typedef`’ов внутри них: `remove_const`, `remove_reference`.

Правила вывода типов для шаблонов. Отбрасывание ссылок при выводе типа. Разбор случаев со ссылками и константами.

Параметры шаблонов, не являющиеся типами (пример: массив константной длины). Параметры шаблонов, являющиеся шаблонами (“template template parameters”).

Шаблоны с переменным количеством аргументов (variadic templates). Синтаксис использования. “Откусывание” шаблонных аргументов по одному. Оператор “`sizeof...`”.

Функциональные классы и функциональные объекты (функторы), схема использования. Компараторы. Пример: компаратор в `std::sort`. Стандартные компараторы (`std::less`, `std::greater`, `std::equal` и т. п.), их реализация.

Curiously Recurring Template Pattern (CRTP).

## 7. Исключения (exceptions)

Общая идея, мотивировка использования исключений, оператор `throw` и конструкция `try...catch`. Примеры стандартных операторов, генерирующих исключения.

Разница между исключениями и ошибками времени выполнения. Ошибки, не являющиеся исключениями, и исключения, не являющиеся ошибками.

Правила ловли и повторного бросания исключений, приведения типов при ловле исключений. Ловля всех исключений. Правила выбора блока `catch` компилятором в случае, когда подходят разные блоки.

Копирование при бросании и ловле исключений, исключения и наследование. Особенности перехвата исключений по значению и по ссылке, по ссылке на базовый класс.

Спецификации исключений в старом стиле и их проблемы, `unexpected exceptions` (неожиданные исключения). спецификации исключений в стиле C++11, оператор и спецификатор `noexcept`. Условный `noexcept`.

Исключения в конструкторах и проблема утечки памяти при исключениях.

Исключения в деструкторах, функция `uncaught_exception`, функции `terminate` и `set_terminate`.

Гарантии безопасности при исключениях: базовая и строгая.

Function-try блоки, их особенности.

## 8. Аллокаторы (allocators)

Placement new, его синтаксис, действие и отличие от обычного new.

Разница между оператором new и функцией `operator new`. Более подробный разбор действия оператора new. Перегрузка new для отдельных классов. Перегрузка глобального new. Определение new с произвольными параметрами. То же самое для операторов delete и delete[]. Пример, когда компилятор неявно вызывает delete с нестандартными параметрами. Поведение delete для полиморфных объектов.

nothrow оператор new, его синтаксис и особенности.

Разбор поведения new в случае нехватки памяти. Функция `new_handler`, функции `set_new_handler` и `get_new_handler`.

Понятие аллокатора. Класс `std::allocator`, его основные методы (`allocate`, `deallocate`, `construct`, `destroy`) и их примерная реализация. Особенности реализации конструкторов и оператора присваивания у стандартного аллокатора.

Класс `std::allocator_traits`, его предназначение и основные методы.

Пример нестандартного аллокатора (`PoolAllocator`), идея реализации его методов. Проблемы с конструктором копирования и оператором присваивания.

## 5. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине (модулю)

Учебная аудитория, оснащенная компьютером и мультимедийным оборудованием (проектор, звуковая система).

## 6. Перечень рекомендуемой литературы

Основная литература

1. Программирование на C++ [Электронный ресурс], Электрон. версия печ. публикации / Н. Дейл, Ч. Уимз, М. Хедингтон. — М., ДМК Пресс, 2007

Дополнительная литература

**7. Перечень ресурсов информационно-телекоммуникационной сети "Интернет", необходимых для освоения дисциплины (модуля)**

Не используются

**8. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень необходимого программного обеспечения и информационных справочных систем (при необходимости)**

ПО для разработки и отладки программ на языке программирования C++.

**9. Методические указания для обучающихся по освоению дисциплины (модуля)**

Выполнение домашнего задания является одной общей работой студента. Необходимо требовать от студентов качественного и понятного программного кода на ряду и в равной степени с правильной работой программы.

**ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)**

**по направлению:** Прикладная математика и информатика  
**профиль подготовки:** Анализ данных в экономике  
Физтех-школа Прикладной Математики и Информатики  
кафедра алгоритмов и технологий программирования  
**курс:** 1  
**квалификация:** бакалавр

Семестр, формы промежуточной аттестации: 1 (осенний) - Дифференцированный зачет

**Разработчик:** И.С. Мещерин, ассистент



## 1. Компетенции, формируемые в процессе изучения дисциплины

Код и наименование компетенции	Индикаторы достижения компетенции
ОПК-1 Способен применять фундаментальные знания, полученные в области физико-математических и (или) естественных наук и использовать их в профессиональной деятельности	ОПК-1.2 Способен строить математические модели, производить количественные расчеты и оценки
ОПК-2 Способен использовать современные информационные технологии и программные средства при решении задач профессиональной деятельности, соблюдая требования информационной безопасности	ОПК-2.2 Знает и умеет применять численные математические методы и прикладное программное обеспечение для решения научных задач в профессиональной области

## 2. Показатели оценивания компетенций

В результате изучения дисциплины «Программирование на языке C++. Продвинутый поток (ФПМИ)» обучающийся должен:

### знать:

- алгоритмы на графах и структуры данных, связанные с ними;
- оценки сложности стандартных алгоритмов;
- стандартные алгоритмы на графах и используемые структуры данных, подходы к модификации классических алгоритмов;
- разнообразные классические задачи в теории графов и асимптотические сложности их решений.

### уметь:

- формулировать задачи в терминах изученных теорий, выбирать подходящий алгоритм для поставленной задачи;
- разрабатывать комбинации алгоритмов для решения поставленной задачи;
- оценивать сложности алгоритмов, их модификаций и комбинаций, в том числе с помощью амортизационного анализа;
- выбирать подходящие структуры данных для конкретной задачи;
- реализовывать алгоритм в обобщенной форме на языке программирования c++;
- реализовывать стандартные алгоритмы на графах и структуры данных на языке программирования C++.

### владеть:

- методами декомпозиции задач в области информационных технологий и построения единого решения с использованием изученных алгоритмов;
- методами оценки сложности алгоритмов, их модификаций и комбинаций.

## 3. Перечень типовых (примерных) вопросов, заданий, тем для подготовки к текущему контролю

1. Что такое выражения, операторы? Для чего предназначены и что по стандарту делают следующие операторы: тернарный оператор, оператор “запятая”, унарная звездочка, унарный амперсанд, операторы “точка” и “стрелочка”, двойное двоеточие, префиксный и постфиксный инкремент, бинарные & и &&, операторы простого и составного присваивания, операторы << и >>?
2. Объясните идею ссылок (references). Для чего они нужны, чем они отличаются от указателей? Что такое “передача аргументов по ссылке и по значению”? Как в C++03 реализовать функцию swap?
3. Что такое конструкторы, деструкторы, для чего они нужны, каков синтаксис их определения? Те же вопросы про конструктор копирования и оператор присваивания.
4. Что такое наследование? В чем разница между приватным и публичным наследованием?
5. Что такое полиморфизм? Приведите пример полиморфизма в C++.
6. Что такое исключения? Как пользоваться механизмом обработки исключений? Как бросить, поймать исключение? Приведите какой-нибудь пример.

7. Что такое компараторы? Что такое функциональные объекты? Приведите хоть один пример.
8. Что такое умные указатели? Для решения каких проблем они нужны? Приведите пример использования.
9. Что такое лямбда-выражения, каков их синтаксис? Приведите хоть один пример использования.
10. Что такое SFINAE? Приведите хотя бы один пример (на уровне идей, можно без реализации).

#### **4. Перечень типовых (примерных) вопросов и тем для проведения промежуточной аттестации обучающихся**

1. Объясните понятия `compilation error`, `runtime error` и `undefined behaviour`. Приведите по паре примеров того, другого и третьего.
2. Что такое стековая память и динамическая память? Что такое `stack overflow`? Зачем нужен оператор `new`? Что такое утечки памяти? Что такое сборка мусора?
3. Что такое класс, структура, в чем разница между ними? Что такое поля, методы, модификаторы доступа, инкапсуляция?
4. Что такое перегрузка функций? Что такое перегрузка операторов? Приведите примеры операторов, которые можно и нельзя перегружать. Приведите пример перегрузки хоть какого-нибудь оператора (напишите сигнатуру его перегрузки).
5. Что такое виртуальные функции, чисто виртуальные функции, абстрактные классы? Что такое виртуальное наследование?
6. Что такое шаблоны? Что такое инстанцирование шаблонов, специализация шаблонов? Приведите пример каких-нибудь шаблонов из STL, обладающих специализацией.
7. Рассмотрим контейнеры `std::vector`, `std::list`, `std::deque`. Каковы основные операции, предоставляемые ими, и скорость работы этих операций?
8. Рассмотрим контейнеры `std::map`, `std::set`, `std::unordered_map`, `std::unordered_set`. Каковы основные операции, предоставляемые ими, и скорость работы этих операций?
9. Что такое итераторы? Какие виды итераторов существуют? Зачем они вообще нужны? Какие итераторы поддерживает каждый из контейнеров, упомянутых в предыдущих двух вопросах?
10. Для чего нужна `move`-семантика? Расскажите в общих чертах, что это такое. Как правильно реализовать функцию `swap` в C++11?
11. Что такое аллокаторы? Какова общая идея класса `std::allocator`, как и для чего он используется?
12. Напишите вычисление  $n$ -го числа Фибоначчи (в пределах `long long`) на этапе компиляции, не используя слова `constexpr`.

#### **Критерии оценивания**

отлично

- 10 Полностью и вовремя решены все задачи без ошибок. Продемонстрирован грамотный подход к решению задач, реализованы оптимальные алгоритмы, код оформлен в едином удобочитаемом стиле
- 9 Полностью и вовремя решены все задачи без ошибок. Продемонстрирован грамотный подход к решению задач, реализованы оптимальные алгоритмы
- 8 Полностью и вовремя решены все задачи без ошибок. Продемонстрирован грамотный подход к решению задач

хорошо

- 7 Полностью решены все задачи. Допущены несущественные ошибки.
- 6 Полностью решено большинство задач. В некоторых задачах допущены и не исправлены ошибки, либо некоторые задачи решены частично.
- 5 Полностью решено две трети задач. В некоторых задачах допущены и не исправлены ошибки, либо некоторые задачи решены частично.

удовлетворительно

- 4 Полностью решено более половины задач. В остальных задачах допущены и не исправлены ошибки, либо некоторые задачи решены частично.

3 Полностью решено более половины задач.

неудовлетворительно

2 Решено менее половины задач.

1 Не решено ни одной задачи.

**5. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности**

Во время проведения дифференцированного зачета обучающиеся могут пользоваться программой дисциплины. При проведении устного дифференцированного зачёта обучающемуся предоставляется 30 минут на подготовку.