Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
Московский государственный университет имени М.В. Ломоносова
Факультет Вычислительной математики и кибернетики
Кафедра Алгоритмических языков

УТВЕРЖДАЮ
Декан факультета ВМК МГУ
_____ /И.А.Соколов/
«__1__» _____ 2023 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ)

Наименование дисциплины (модуля):

Объектно-ориентированное программирование

Уровень высшего образования:

бакалавриат

Направление подготовки (специальность):

02.03.02 Фундаментальная информатика и информационные технологии

Направленность (профиль) ОПОП:

дисциплина относится к базовой части программы

Форма обучения:
очная

Москва 2023

Рабочая программа дисциплины (модуля) разработана в соответствии с самостоятельно установленным МГУ образовательным стандартом (ОС МГУ) для реализуемых основных профессиональных образовательных программ бакалавриата, магистратуры, реализуемых последовательно по схеме интегрированной подготовки по направлениям 02.03.02, 02.04.02 «Фундаментальная информатика и информационные технологии» в редакции приказа МГУ от 30 декабря 2016г.

1. Место дисциплины (модуля) в структуре ОПОП ВО

Дисциплина относится к дисциплинам базовой части ОПОП ВО и является обязательной для освоения в 3-ем семестре обучения.

2. Входные требования для освоения дисциплины (модуля), предварительные условия:

Учащиеся должны владеть знаниями по алгоритмам и алгоритмическим языкам, архитектуре ЭВМ и языку Ассемблера, основам программирования на языке Си в объеме, соответствующем программе первого года обучения основных образовательных программам бакалавриата по укрупненным группам направлений и специальностей 01.00.00 «Математика и механика», 02.00.00 «Компьютерные и информационные науки».

3. Результаты обучения по дисциплине (модулю), соотнесенные с требуемыми компетенциями выпускников.

Компетенции выпускников, формируемые (полностью или частично) при реализации дисциплины (модуля):

ОПК-2 - способность применять в профессиональной деятельности современные языки программирования и языки баз данных, методологии системной инженерии, системы автоматизации проектирования, электронные библиотеки и коллекции, сетевые технологии, библиотеки и пакеты программ, современные профессиональные стандарты информационных технологий;

ОПК-3 - способность к разработке алгоритмических и программных решений в области системного и прикладного программирования, математических, информационных и имитационных моделей, тестов и средств тестирования систем и средств на соответствие стандартам и исходным требованиям;

ПК-2 - способность понимать, совершенствовать и применять современный математический аппарат, фундаментальные концепции и системные методологии, международные и профессиональные стандарты в области информационных технологий;

ПК-6 - способность эффективно применять базовые математические знания и информационные технологии при решении проектно-технических и прикладных задач, связанных с развитием и использованием информационных технологий;

ПК-7 - способность разрабатывать и реализовывать процессы жизненного цикла информационных систем, программного обеспечения, сервисов систем информационных технологий, а также методы и механизмы оценки и анализа функционирования средств и систем информационных технологий;

ПК-8 - способность применять на практике международные и профессиональные стандарты информационных технологий, современные парадигмы и методологии, инструментальные и вычислительные средства.

Планируемые результаты обучения по дисциплине (модулю):

<u>знать</u>

- основные парадигмы программирования;
- основные понятия и концепции объектно-ориентированной парадигмы программирования;
- основные элементы объектно-ориентированного анализа;

- связь языка C++ с языком C;
- понятие абстрактного типа данных;
- понятие класса в языке C++, управление доступом к членам класса;
- понятие пространства имен, разрешение области видимости;
- специальные функции – конструкторы и деструктор;
- понятие ссылки на объект;
- понятие квалификатора const в C++;
- статические члены класса;
- понятие о функциях - друзьях класса, «законы» дружбы;
- понятие перегрузки функций;
- понятие полиморфизма и его виды (статический, динамический, параметрический);
- основные понятия наследования.

уметь

- применять на практике основные методы объектно-ориентированной парадигмы программирования;
- применять на практике компьютерные технологии для решения различных задач в объектно-ориентированном стиле;
- извлекать полезную научно-техническую информацию из электронных библиотек и реферативных журналов;
- создавать алгоритмические модели типовых задач, проводить спецификацию задачи, реализовывать программы на языке C++, оценивать сложность полученных алгоритмов;
- публично представить собственные и известные научные результаты;
- самостоятельно разрабатывать алгоритмы для решения задач системного и прикладного программирования.

владеть

- навыками решения практических задач на языке C++;
- методами объектно-ориентированного программирования;
- навыками решения практических задач на основе использования объектно-ориентированной парадигмы программирования.

иметь опыт:

- разработки компонентов программного обеспечения на основе использования объектно-ориентированной парадигмы программирования;
- решения практических задач, связанных с использованием языка программирования Си++.

4. Формат обучения

Используются традиционные технологии проведения лекций и семинарских занятий в аудиториях с использованием меловой доски, а также чтение лекций с использованием слайдов. При чтении лекций наиболее важные элементы лекций обсуждаются с аудиторией в режиме «вопрос-ответ». Семинары по данной дисциплине проводятся как в формате аудиторных занятий, так и в виде практических занятий в компьютерном классе.

Дополнительно, на семинаре студенты выполняют небольшие практические задания по тематике данного семинара. На каждом занятии проводится обсуждение домашних заданий, а также все студенты имеют возможность задать преподавателям свои вопросы по изучаемой теме.

Лектор курса и преподаватели, ведущие практические занятия, периодически проводят консультации по дисциплине.

5. Объем дисциплины (модуля) составляет 3 зачетные единицы, всего 108 часов,

в том числе 36 академических часов, отведенных на контактную работу обучающихся с преподавателем (лекции – 18 часов, семинары – 18 часов), 72 академических часа на самостоятельную работу обучающихся.

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

Целью освоения данного курса является изучение основных концепций и методов объектно-ориентированного программирования, а также изучение языка программирования C++, в котором эти концепции и методы воплощены наиболее полно.

В частности, ставятся следующие задачи:

- изучить основные принципы объектно-ориентированной парадигмы программирования, как наиболее распространенной и востребованной в настоящее время;
- изучить основные возможности объектно-ориентированного языка программирования C++;
- изучить основные методы программирования на языке C++;
- получить навыки практического программирования на языке C++.

Изучение опирается на знания, полученные студентами в результате прослушивания курсов «Алгоритмы и алгоритмические языки», «Архитектура ЭВМ и язык ассемблера» и «Основы программирования». Курс «Объектно-ориентированное программирование» будет полезен для успешного освоения курса «Системы программирования» (читается в следующем семестре), в котором предлагаемый материал получит свое развитие.

Наименование и краткое содержание разделов и тем дисциплины (модуля), **Форма промежуточной аттестации по дисциплине (модулю)**	**Всего (часы)**	В том числе			
		Контактная работа (работа во взаимодействии с преподавателем) Виды контактной работы, часы			**Самостоятельная работа обучающегося, часы** *(виды самостоятельной работы – эссе, реферат, контрольная работа и пр. –)*
		Занятия лекционного типа*	Занятия семинарского типа*	**Всего**	

1.Объектно-ориентированное программирование (ООП) – новая технология (парадигма) программирования. Парадигмы программирования. Процессно-ориентированный и объектно-ориентированный подходы к программированию. Основные свойства языка, поддерживающего ООП: абстракция, инкапсуляция, наследование, полиморфизм.	4	2	0	2	2
2.Объектно-ориентированный анализ и объектно-ориентированное проектирование. Понятие объекта. Выделение используемых объектов, фиксация связей между объектами, фиксация методов обмена сообщениями между объектами.	14	2	4	6	8
3. История возникновения и развития языка С++. Язык С++ в сравнении с языком Си. Обзор средств языка С++, реализующих механизмы ООП. Пространства имен, разрешение области видимости. Понятие класса в языке С++. Описание класса: члены-данные и члены-функции. Управление доступом к членам класса – public, private. Объявления и описания функций-членов класса; эффект inline.	6	2	0	2	4
4 Специальные функции – конструкторы и деструктор. Перегрузка конструкторов. Конструктор умолчания. Конструктор преобразования.	14	2	4	6	8
5.Текущий контроль успеваемости: контрольная работа № 1 (реферат).	5	0	1	1	4
6. Понятие ссылки на объект. Передача параметров в функции по ссылке. Возврат результата из функции по ссылке.	10	2	2	4	6

7. Конструктор копирования, генерация конструктора копирования по умолчанию, определение конструктора копирования. Конструктор копирования и операция присваивания: содержательная связь и различие. Указатель this. Временные переменные. Члены класса – объекты другого класса. Порядок вызова конструкторов, список инициализации.	8	2	2	4	4
8.Использование квалификатора const в C++. Квалификатор mutable. Операторы new и delete. Работа с динамической памятью. Статические члены класса.	8	2	2	4	4
9. Перегрузка функций. Аргументы со значениями по умолчанию. Перегрузка и неоднозначность. Функции - друзья класса. «Законы» дружбы.	9	2	1	3	6
10.Текущий контроль успеваемости: контрольная работа № 2.	4	0	2	2	2
11. Алгоритм поиска оптимально отождествляемой (best-matching) функции.	4	2	0	2	2
12.Текущий контроль успеваемости: коллоквиум	4	0	0	0	4
13.Промежуточная аттестация: устный экзамен.	18	0	0	0	18
Итого	108	18	18	36	72

7. Фонд оценочных средств (ФОС) для оценивания результатов обучения по дисциплине (модулю)

7.1. Типовые контрольные задания или иные материалы для проведения текущего контроля успеваемости.

Контрольная работа № 1 (в форме реферата).

Анализ и проектирование с использованием объектно-ориентированной парадигмы программирования

Постановка задачи:

Выбрать для исследования произвольную предметную область (ПО). Выявить в выбранной ПО некоторое количество сущностей (понятий) – будущие классы. Для каждой из сущностей определить признаки, свойства - будущие члены-данные класса, особенности создания, копирования и уничтожения – будущие конструкторы и деструктор, возможное поведение сущности - будущие методы класса, ее использование – будущие внешние функции, а также связи с другими сущностями данной ПО. Представить результат анализа и проектирования произвольным образом - в виде словесного описания, схем, диаграмм. Примеры возможных ПО: банк, магазин, университет, спортивный клуб. (Впоследствии предполагается реализация предложенной модели ПО на языке С++.)

Контрольная работа № 2 (проводится на компьютере).

Вариант

Дополнить программу, не изменяя текст функции main(), чтобы функция main() работала в соответствии со своим описанием, приведенным в комментариях. В описании классов не использовать открытые нестатические члены – данные.

```
int main(){
        Ball gb ("green",20),  //мяч цвета "green", диаметр – 20
              wb(12), //мяч цвета "white", диаметр – 12
              b(10);  //мяч цвета "white", диаметр – 10
        cout<<"The smallest:"<<smallest(gb,wb,b)<<" end "<<endl;
         //должен быть напечатан       диаметр самого маленького из мячей
         return 0;
}
```

В результате работы программы должно быть напечатано:

```
The smallest: 10 end
 white
 white
 green
```

Вариант письменного коллоквиума (образец)

1. Что такое АТД? Каким образом АТД реализуется в С++?

2. Описать класс myclass так, чтобы:
- все конструкции функции main ( ) были верными,
- явно в классе myclass можно описать не более одного конструктора,
- на экран выдалось **10 20 30**

Нельзя использовать исключения и любые функции досрочного завершения программы.

```
int main () {
        myclass  a1,   a2 = a1,  a3 (a2);
        cout << a1.get ( ) <<  ' ' << a2.get  () << ' ' <<  a3.get ( ) << endl;
        return 0;
}
```

3. Что будет выдано на печать при работе следующей программы?

```
struct S {
        int x;
        S (int n) {   x = n;   printf (" Cons    "); }
        S (const S & a) {  x = a.x;   printf (" Copy    "); }
        ~S ( ) {  printf ("Des   "); }
};

S  f ( S  y ) {  y = S (3);  return y; }

int main ( ) {
        S s (1);
        f (s);
  printf ("%d   ",  s.x);
  return 0;
}
```

4. Есть ли ошибки в приведенной ниже программе на Си++? Если есть – объясните, в чем они заключаются. Если нет – прокомментируйте работу программы.

```
class Flower{
        int num;
        int height;
        char* name;
        public:  Flower(int n=10, int h, char* nm = "Rose"):   num(n), height(h) {name=nm; }
                 void  ~Flower() {delete[] name;}
   };

class Garden{
       Flower& f;
       public: void add_flower  (Flower& f1) {f=f1;}
   };

int main(){
     Garden g;
     Flower f;
     return 0;
  }
```

5. Дать определение деструктора. Привести примеры двух различных ситуаций, в которых вызывается деструктор.

6. Добавить (если нужно) в класс A сл.слова «**const**» , так, чтобы заданный фрагмент программы был верным.

```
class A {
        int i;
public:
        A (int x) { i = x; }
        A (A & y) { i = y.i; }
```

```
        const A f (const A & z) { cout << endl;  return *this;}
};
const A  t1 ( ) {
        const A a = 5;
        return a.f ( a );
}
```

7. Назвать три разных ситуации, когда **автоматически** вызывается конструктор копирования.

8. Описать конструктор для некоторого класса A таким образом, чтобы были выполнены следующие условия:
а) это единственный явно описанный конструктор класса A,
б) справедливы следующие описания объектов класса A

```
A a;  A b(1);  A c(1, 2);  A d('1', 1);
```

9. Для каждого вызова перегруженной функции с одним параметром укажите, какая функция и на каком шаге алгоритма будет выбрана.

```
int f (int a = 0) { return a; }
int f (double a) { return a; }
int main() { short int s;
             int i;
             bool b;
             enum e {A, B, C};
             float f1 = 1.0f;
  f();
  f(s);
  f(f1);
  f(b);
  f(A);
}
```

7.2. Типовые контрольные задания или иные материалы для проведения промежуточной аттестации.

Вопросы к экзамену

1. Парадигмы программирования. Объектно-ориентированное программирование (ООП) -- новая технология (парадигма) программирования.
2. ООП-анализ.
3. Пространства имен в языке Си++.
4. Основные свойства языка, поддерживающего ООП.
5. Понятие класса и объекта. Описание класса.
6. Управление доступом к членам класса -- public, private.
7. Операции . и ->, символ ::, указатель this.
8. Объявления и описания функций-членов класса; эффект inline.
9. Специальные функции -- конструкторы и деструктор.
10. Перегрузка конструкторов.
11. Конструктор копирования.

12. Ссылки и указатели в Си++.
13. Операторы new и delete.
14. Статические члены класса.
15. Константные функции-члены.
16. Друзья класса, "законы" дружбы. Сравнение функции-члена и функции-друга: описание, вызов.
17. Перегрузка функций. Перегрузка и неоднозначность.
18. Функции с параметрами по умолчанию.
19. Алгоритм поиска оптимально отождествляемой (best-matching) функции. Общая характеристика.

Типовые задачи для экзамена

Задача №1.
Есть ли ошибки в приведенном фрагменте программы на С++? Если есть, то объясните, в чем они заключаются. Вычеркните ошибочные конструкции (если они есть). Что будет выдано в стандартный канал вывода при вызове функции main()?

```
void f1 ( ) { cout << 0; }

class X {
        int i;
       double t;
        X ( int k = 0) { i = k;     t = k / 10;   cout << 1; }
      public:
        X (double r ) { i = 0;    t = r;     cout << 2; }
         X (int k, double r) { i = k;    t = r;   cout << 4;}
         void f1 (int a) {i = a;  t = a / 2.0;}
};
int main () {   X a (1);
                X b (2.5);
                 X c;
                 X d (1.5, 5);
                 f1 ( 1 );
                b = d;
                 return 0;
                 }
```

Задача №2.
Даны описания структуры, переменной и функции:

```
struct str {
    int a , b;};
int i = sizeof(str);
int f( str s) {
    return 0;
      }
```

Дополните описание структуры str (не изменяя описание функции f) так, чтобы только описание f стало ошибочным.

Задача №3.
Что будет выдано на печать при работе следующей программы?

```
#include <iostream>
struct S {
        int x;
        S (int n) {   x = n;   printf (" Cons    "); }
        S (const S & a) {  x = a.x;   printf (" Copy    "); }
        ~S ( ) {  printf ("Des   "); }
};

S  f ( S & y ) {  y = S (5);  return y; }
int main ( ) {
        S s (8);
        f (s);
        printf ("%d   ",  s.x);
        return 0;
}
```

Экзаменационный билет состоит из двух теоретических вопросов и задачи, например:

1.Ссылки и указатели в Си++.
2.Друзья класса, "законы" дружбы. Сравнение функции-члена и функции-друга: описание, вызов.
3. Даны описания структуры, переменной и функции:

```
struct str {
      int a , b;};

int i = sizeof(str);

int f( str s) {
     return 0;
        }
```

Дополните описание структуры str (не изменяя описание функции f) так, чтобы только описание f стало ошибочным.

Методические материалы для проведения процедур оценивания результатов обучения

Используется дифференцированная система оценки знаний и навыков. Оценка основывается на:

- контроле посещаемости занятий;
- результатах сдачи контрольных работ;
- результатах сдачи коллоквиума;
- результате сдачи экзамена.

При этом, в порядке исключения, учитывая высокую оценку работы в семестре, часть студентов может получить «отличные» и «хорошие» оценки за экзамен без сдачи устного экзамена.

ШКАЛА И КРИТЕРИИ ОЦЕНИВАНИЯ результатов обучения (РО) по дисциплине (модулю)

Оценка / РО и соответствующие виды оценочных средств	2	3	4	5
Знания *Коллоквиум, Экзамен*	Отсутствие знаний	Фрагментарные знания	Общие, но не структурированные знания	Сформированные систематические знания
Умения *Контрольные работы, коллоквиум*	Отсутствие умений	В целом успешное, но не систематическое умение	В целом успешное, но содержащее отдельные пробелы умение (допускает неточности непринципиального характера)	Успешное и систематическое умение
Навыки (владения, опыт деятельности) *Экзамен*	Отсутствие навыков (владений, опыта)	Наличие отдельных навыков (наличие фрагментарного опыта)	В целом, сформированные навыки (владения), но используемые не в активной форме	Сформированные навыки (владения), применяемые при решении задач

Соответствие результатов обучения и компетенций, в развитии которых участвует дисциплина (модуль)

Результаты обучения	Компетенция, с частичным формированием которой связано достижение результата обучения
Знать: • основные парадигмы программирования; • основные понятия и концепции объектно-ориентированной парадигмы программирования; • основные элементы объектно-ориентированного анализа; • связь языка C++ с языком C; **Уметь** • применять на практике основные методы объектно-ориентированной парадигмы программирования; • применять на практике компьютерные технологии для решения различных задач в объектно-ориентированном стиле; • извлекать полезную научно-техническую информацию из электронных библиотек и реферативных журналов; • публично представить собственные и известные научные результаты;	ОПК-2

Владеть • методами объектно-ориентированного программирования; **Иметь опыт:** • разработки компонентов программного обеспечения на основе использования объектно-ориентированной парадигмы программирования;	
Знать: • связь языка С++ с языком С; • понятие абстрактного типа данных; • понятие класса в языке С++, управление доступом к членам класса; • понятие пространства имен, разрешение области видимости; • специальные функции – конструкторы и деструктор; • понятие ссылки на объект; • понятие квалификатора const в С++; • статические члены класса; • понятие о функциях - друзьях класса, «законы» дружбы; • понятие перегрузки функций; • понятие полиморфизма и его виды (статический, динамический, параметрический); • основные понятия наследования. **Уметь:** • применять на практике компьютерные технологии для решения различных задач в объектно-ориентированном стиле; • создавать алгоритмические модели типовых задач, проводить спецификацию задачи, реализовывать программы на языке С++, оценивать сложность полученных алгоритмов; • самостоятельно разрабатывать алгоритмы для решения задач системного и прикладного программирования. **Владеть**: • навыками решения практических задач на языке С++; • навыками решения практических задач на основе использования объектно-ориентированной парадигмы программирования. **Иметь опыт:** • разработки компонентов программного обеспечения на основе использования объектно-ориентированной парадигмы программирования; • решения практических задач, связанных с использованием языка программирования Си++.	ОПК-3
Знать: • основные парадигмы программирования; • основные понятия и концепции объектно-ориентированной парадигмы программирования; • основные элементы объектно-ориентированного анализа;	ПК-2

Уметь: • применять на практике основные методы объектно-ориентированной парадигмы; • применять на практике компьютерные технологии для решения различных задач в объектно-ориентированном стиле; • извлекать полезную научно-техническую информацию из электронных библиотек и реферативных журналов; • самостоятельно разрабатывать алгоритмы для решения задач системного и прикладного программирования. **Владеть**: • навыками решения практических задач на языке С++; • навыками решения практических задач на основе использования объектно-ориентированной парадигмы программирования. **Иметь опыт:** • разработки компонентов программного обеспечения на основе использования объектно-ориентированной парадигмы программирования;	
Знать: • связь языка С++ с языком С; • понятие абстрактного типа данных; • понятие класса в языке С++, управление доступом к членам класса; • понятие пространства имен, разрешение области видимости; • специальные функции – конструкторы и деструктор; • понятие ссылки на объект; • понятие квалификатора const в С++; • статические члены класса; • понятие о функциях - друзьях класса, «законы» дружбы; • понятие перегрузки функций; • понятие полиморфизма и его виды (статический, динамический, параметрический); • основные понятия наследования. **Уметь:** • создавать алгоритмические модели типовых задач, проводить спецификацию задачи, реализовывать программы на языке С++, оценивать сложность полученных алгоритмов; • самостоятельно разрабатывать алгоритмы для решения задач системного и прикладного программирования. **Владеть**: • навыками решения практических задач на языке С++; **Иметь опыт:** • решения практических задач, связанных с использованием языка программирования Си++.	ПК-6

Знать • основные парадигмы программирования; • основные понятия и концепции объектно-ориентированной парадигмы программирования; • основные элементы объектно-ориентированного анализа; **Уметь** • применять на практике основные методы объектно-ориентированной парадигмы программирования; • применять на практике компьютерные технологии для решения различных задач в объектно-ориентированном стиле; • извлекать полезную научно-техническую информацию из электронных библиотек и реферативных журналов; • самостоятельно разрабатывать алгоритмы для решения задач системного и прикладного программирования. **Владеть** • навыками решения практических задач на языке C++; • методами объектно-ориентированного программирования; • навыками решения практических задач на основе использования объектно-ориентированной парадигмы программирования. **Иметь опыт:** • разработки компонентов программного обеспечения на основе использования объектно-ориентированной парадигмы программирования;	ПК-7
Знать • основные парадигмы программирования; • основные понятия и концепции объектно-ориентированной парадигмы программирования; • основные элементы объектно-ориентированного анализа; **Уметь** • применять на практике основные методы объектно-ориентированной парадигмы программирования; • применять на практике компьютерные технологии для решения различных задач в объектно-ориентированном стиле; • публично представить собственные и известные научные результаты; • самостоятельно разрабатывать алгоритмы для решения задач системного и прикладного программирования. **Владеть** • методами объектно-ориентированного программирования; • навыками решения практических задач на основе использования объектно-ориентированной парадигмы программирования. **Иметь опыт:** • разработки компонентов программного обеспечения на основе	ПК-8

использования объектно-ориентированной парадигмы программирования;	

8. Ресурсное обеспечение:

Основная литература:

1. И. А. Волкова, А. В. Иванов, Л. Е. Карпов. Основы объектно-ориентированного программирования. Язык программирования C++. Учебное пособие для студентов 2 курса. М.: Издательский отдел факультета ВМК МГУ, 2011.
Электронная версия: http://cmcmsu.no-ip.info/download/cpp.base.oop.pdf
2. Ю.С. Корухова. Сборник задач и упражнений по языку C++. Учебное пособие для студентов-бакалавров II курса, обучающихся по направлению «Информационные технологии». М.: Издательский отдел факультета ВМК МГУ, 2009
Электронная версия: http://cmcmsu.no-ip.info/download/korukhova.cpp.tasks.pdf
3. А.В.Столяров. Введение в язык C++. Учебное пособие. 3 изд. М.МАКС Пресс, 2012.
Электронная версия:http://www.stolyarov.info/books/pdf/cppintro3.pdf

Дополнительная литература:

1. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++. Третье издание, М. ООО «И.Д.Вильямс», 2017.
2. Страуструп Б. Язык программирования C++. Специальное изд./Пер. с англ. - М.: "Бином", 2015.
3. Страуструп Б. Программирование: принципы и практика использования C++.: Пер. с англ. – М. ООО «И.Д.Вильямс», 2016.
4. Пол А. "Объектно-ориентированное программирование на Си++" – второе издание, М., Бином, 2001.
5. Шилдт Г.. Самоучитель C++. – СПб, BHV, 2006.
6. Страуструп Б. Дизайн и эволюция C++. Пер. с англ. – М.:ДМК Пресс, 2014.

Кроме того, студентам предлагается искать дополнительную информацию на сайтах, посвященных объектно-ориентированному программированию на языке C++, в частности на сайте комитета по стандартизации языка C++ (The C++ Standards Committee) http://www.open-std.org/JTC1/SC22/WG21/

Материально-техническое обеспечение:

Для проведения аудиторных лекционных и семинарских занятий необходима аудитория с партами и меловой доской.

Для проведения практических занятий необходимо наличие компьютерного класса с возможностью работы с компилятором языка C++.

9. Язык преподавания.

Язык преподавания дисциплины — русский.

10. Преподаватель (преподаватели): Доцент факультета ВМК МГУ Полякова И.Н.

11. Авторы программы: Доцент факультета ВМК МГУ Полякова И.Н.
Ассистент факультета ВМК МГУ Кузина Л.Н.