Федеральное государственное бюджетное образовательное
учреждение высшего образования
Московский государственный университет имени М.В.Ломоносова
Филиал Московского государственного университета имени М.В.Ломоносова
в городе Сарове

УТВЕРЖДАЮ

Директор филиала МГУ в городе Сарове

_____ /В.В. Воеводин/

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Наименование дисциплины:

«Объектно-ориентированный подход в современных языках программирования»

Уровень высшего образования:

магистратура

Направление подготовки / специальность:

02.04.02 "Фундаментальная информатика и информационные технологии" (3++)

Направленность (профиль)/специализация ОПОП:

Суперкомпьютерные технологии и фундаментальная информатика

Форма обучения:

очная

Саров 2022

Рабочая программа дисциплины (модуля) разработана в соответствии с самостоятельно установленным МГУ образовательным стандартом (ОС МГУ) для реализуемых основных профессиональных образовательных программ высшего образования по направлению подготовки 02.04.02 "Фундаментальная информатика и информационные технологии" программы магистратуры - приказ МГУ 30 августа 2019 года № 1054 (в редакции приказа МГУ от 11 сентября 2019 года № 1109)

1. ЦЕЛИ И ЗАДАЧИ ДИСЦИПЛИНЫ

1.1. Целями дисциплины являются:

– знакомство с современными парадигмами и языками программирования;
– изучение основных концепций и методов объектно-ориентированного программирования, а также ключевых концепций императивного и функционального программирования и их интеграции к ООП-подходу в программировании;
– знакомство с инструментами разработки ООП-программ.

1.2. Задачи дисциплины:

1) изучить возможности, назначение, состав и принципы использования современных объектно-ориентированных языков и систем программирования (на примере языков C++ и Фортран);

2) изучить принципы объектно-ориентированной, а также императивной и функциональной парадигм программирования в приложении к ООП-подходу;

3) изучить основные возможности и методы программирования на объектно-ориентированном языке программирования C++;

4) изучить основные возможности и методы программирования на языке программирования Фортран и научиться использовать языковые средства Фортрана для построения программ в ООП-стиле;

5) проиллюстрировать применение объектно-ориентированного языка программирования C++ на примере разработки интерпретатора для модельных вычислений.

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОПОП

А. Информация об образовательном стандарте и учебном плане.

Тип Стандарта: магистр филиала МГУ в Сарове, учебный план магистратуры;

- **направление подготовки** «Фундаментальная информатика и информационные технологии»,
- **наименование учебного плана** «Суперкомпьютерные технологии и фундаментальная информатика»

Б. **Место дисциплины в образовательном стандарте и учебном плане:**

- вариативная часть, дисциплина по выбору студента;
- курс – 2.
- семестр – 4.

В. Перечень дисциплин, которые должны быть освоены для начала освоения данной дисциплины: Учащиеся должны владеть знаниями по математическому анализу, линейной алгебре, дифференциальным уравнениям, дискретной математике, программированию на языках С/С++ и ФОРТРАН, алгоритмам и структурам данных.

Г. Общая трудоемкость (в ак. часах и зачетных единицах): 72 ак. ч., 2 зач. ед.

3. ПЕРЕЧЕНЬ КОМПЕТЕНЦИЙ, ФОРМИРУЕМЫХ В РЕЗУЛЬТАТЕ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Компетенции	**Результаты обучения**
ПК-4	**Знать:** Виды и инструменты имитационного и информационного моделирования, особенности их применения. Типовые алгоритмы, используемые при имитационном и информационном моделирования. **Уметь:** строить имитационные и информационные модели, создавать алгоритмы для проведения имитационного и информационного моделирования; оценивать и интерпретировать полученные результаты имитационного и информационного моделирования;. **Владеть:** опыт проведения имитационного и информационного моделирования реального процесса.
ПК-5	**Знать:** Типовые алгоритмы, протоколы, вычислительные модели и модели данных по теме выполняемых работ. **Уметь:** разрабатывать алгоритмы, протоколы, вычислительные модели и модели данных по теме выполняемых работ, оценивать их эффективность. **Владеть:** Опытом разработки алгоритмов, протоколов, вычислительных моделей и моделей данных для реализации функций и сервисов систем информационных технологий.
МПК-2	**Знать:** масштабируемые параллельные методы и алгоритмы, используемые при проведении крупномасштабного

	математического моделирования и обработки данных на суперкомпьютерных системах; **Уметь:** разрабатывать и реализовывать масштабируемые параллельные методы и алгоритмы для проведения крупномасштабного математического моделирования и обработки данных на суперкомпьютерных системах; **Владеть:** навыками построения, параллельной реализации и исследования моделей и методов распределенной обработки информации.
МПК-3	**Знать:** основные методы и подходы для оптимизации последовательных и параллельных программ; **Уметь:** оценивать эффективность распределенных алгоритмов; **Владеть:** навыками использования современных инструментальных средств для профилирования и анализа производительности параллельных программ.
МПК-4	**Знать:** способы исследования эффективности функционирования многопроцессорных вычислительных систем **Уметь:** Выполнять теоретическое исследование и экспериментальный анализ эффективности функционирования и методов организации вычислений для многопроцессорных вычислительных систем **Владеть:** Методами организации вычислений на многопроцессорных вычислительных системах; методами масштабируемости параллельных программ.

4. СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

Общая трудоемкость дисциплины составляет 3 зачетные единицы или 108 ч., из них:

- лекции – 36 часов;
- семинары – 0 часов;
- лабораторная работа – 0 часов;
- самостоятельная работа – 36 часов.

Форма контроля – экзамен.

4.1. Распределение трудоемкости по разделам и темам, а также формам проведения занятий с указанием форм текущего контроля и промежуточной аттестации:

№ п/п	Раздел Дисциплины	Виды учебной работы, включая самостоятельную работу студентов и трудоемкость (в часах)		Формы текущего контроля успеваемости *(по неделям семестра)*
		лекции	самост. работа	
1	Введение. Основные принципы	4	4	Самостоятельная работа,

	ООП: абстракция, инкапсуляция, наследование, полиморфизм			индивидуальный опрос, автоматическое тестирование
2	Концепция абстрактных типов данных и ее реализация в C++. Объектно-ориентированный анализ предметной области. Отображение сущностей предметной области в классы. Различия между агрегацией и наследованием. ER-диаграммы	4	4	Самостоятельная работа, индивидуальный опрос
3	Наследование. Правила наследования. Видимость при наследовании. Иерархия классов. Динамическая информация о типе	4	4	Самостоятельная работа, индивидуальный опрос
4	Полиморфизм. Виды полиморфизма в C++. Статический полиморфизм. Динамический полиморфизм. Виртуальные функции. Абстрактные классы	6	6	Самостоятельная работа, индивидуальный опрос, автоматическое тестирование
5	Средства обработки ошибок. Исключения в C++	2	2	Самостоятельная работа, индивидуальный опрос, автоматическое тестирование
6	Особенности множественного наследования. Интерфейсы. Фабрики классов	4	4	Самостоятельная работа, индивидуальный опрос
7	Шаблоны. Шаблонные методы и классы	4	4	Самостоятельная работа, индивидуальный опрос
8	Принципы дизайна и реализации стандартной библиотеки шаблонов C++. Программирование с помощью шаблонов C++. Паттерны метапрограммирования	4	4	Самостоятельная работа, индивидуальный опрос
9	Реализация ООП-подхода в программах на языке Фортран	4	4	Контрольная работа, автоматическое тестирование
	Итого 72	***36***	***36***	

4.2. Содержание дисциплины темам – аудиторная и самостоятельная работа:

Тема 1.

1. ООП – новая технология (парадигма) программирования
2. Основные принципы ООП
3. Обзор средств языка C++, поддерживающих основные механизмы

ООП – новая технология (парадигма) программирования. Понятие объекта в программировании. Процессно-ориентированный и объектно-ориентированный подходы к программированию. Абстракция, инкапсуляция, наследование, полиморфизм.

Обзор средств языка C++, поддерживающих эти механизмы. Язык C++ в сравнении с языком Си. Понятие класса в языке C++. Отображение основных принципов объектно-ориентированного программирования в языке C++ на простейших примерах.

Тема 2.

4. Приёмы декомпозиции

- Объектно-ориентированный анализ предметной области
- Отображение сущностей предметной области в классы
- Преимущества объектной модели

5. Концепция абстрактных типов данных и её реализация в C++
6. Классы и объекты
7. Работа с состоянием объекта: селекторы и модификаторы (*getters/setters*)

Выделение используемых объектов, фиксация связей между объектами, фиксация методов обмена сообщениями между объектами. Пример построения класса объектов. Интерфейс класса.

Концепция абстрактных типов данных и её реализация в C++. Классы как средство создания новых типов. Синтаксис объявления класса. Ограничения доступа к элементам класса. Ключевое слово **`inline`**. Работа с состоянием объекта: селекторы и модификаторы. Процедуры и функции для установки или выдачи значения полей данных класса.

8. Указатели и ссылки на объекты, передача параметров по ссылке
9. Инициализация, конструкторы, копирование, деструкторы

Указатели и ссылки на объекты. Указатель как обобщённый адрес объектов программы. Указатели на константы, константные указатели и константные указатели на константы. Ссылка на объект и её отличие от указателя. Передача параметров по ссылке.

Определения и примеры использования конструкторов, операций копирования и деструкторов. Конструкторы умолчания. Автоматическая генерация конструкторов и деструкторов. Указатель **`this`**. Последовательность выполнения конструкторов и деструкторов для сложных объектов.

Тема 3.

10. Работа с динамической памятью
11. Статические и константные члены классов
12. Пространства именования

Операции **`new`** и **`delete`**. Создание и уничтожение массивов объектов. Плоские и неплоские классы.

Квалификатор **`const`**. Статические методы. Инициализация статических членов класса.

Оператор разрешения области видимости. Видимость пространств именования. Директива **`using`**. Неименованные пространства именования. Отличия классов и пространств именования. Глобальные объекты.

Тема 4.

13. Статический полиморфизм в C++
    - Перегрузка имён
    - Перегрузка функций
    - Друзья классов
    - Перегрузка бинарных операций
    - Перегрузка унарных операций

Статический полиморфизм в C++. Перекрытие имён. Перегрузка функций. Правила работы алгоритма выбора перегруженной функции. Перегрузка бинарных и унарных операций. Пример перегрузки операции индексирования. Особенности перегрузки префиксных и постфиксных операций.

14. Виды отношений между классами
    - ER-диаграммы
    - Ассоциации классов
    - Взаимодействие и иерархия классов
15. Одиночное наследование
16. Виды полиморфизма в C++. Динамический полиморфизм

17. Виртуальные функции

Основные понятия модели Entity-Relationship. Ассоциация классов, агрегация классов, использование одним классом другого класса, инстанцирование класса, наследование одним классом свойств другого класса.

Одиночное (единичное) наследование. Различия между наследованием и агрегацией. Правила наследования. Иерархия классов при наследовании. Решётки смежности. Повторное использование кода. Видимость при наследовании. Правила видимости. Перекрытие имён. Инициализация объектов внутри объектов. Список инициализации. Генеалогические преобразования указателей.

Динамический полиморфизм в C++. Виртуальные функции. Правила создания виртуальных функций. Виртуальные деструкторы.

Тема 5.

18.Исключения в C++

19. Динамическая идентификация типа, средства преобразования типов

Средства обработки ошибок. Исключения в C++. Фундаментальная идея обработки ошибок времени выполнения в программах на C++. Виды реакции на возникновение ошибки в программе. Управление исключительными ситуациями. Действия при возбуждении исключительной ситуации. Вложенные блоки обработчиков исключений. Стандартные исключительные ситуации.

Динамическая идентификация типа, средства преобразования типов. Динамическое и статическое приведение типа. Особенности приведения указателей и ссылок. Другие виды преобразования типов.

Тема 6.

20. Множественное наследование в C++

21. Абстрактные классы

22. Фабрики объектов

23. Реализация виртуальных функций

Множественное наследование в C++. Доступ к членам производного класса. Преобразование указателей. Виртуальные базовые классы. Неоднозначность из-за совпадающих имён в различных базовых классах.

Абстрактные классы. Чистые виртуальные функции. Позднее (динамическое) связывание.

Фабрики объектов. “Виртуальные” конструкторы.

Реализация виртуальных функций с помощью статических таблиц классов.

Темы 7-8.

24.Параметрический полиморфизм в C++. Программирование с помощью шаблонов

25. Основные типы библиотек

26. Критерии проектирования стандартных библиотек

27. Стандартная библиотека языка C++

28. Принципы дизайна и реализация стандартной библиотеки шаблонов C++

29. Основные виды компонентов библиотеки шаблонов STL

Шаблоны функций. Перегрузка шаблонных функций. Шаблоны классов. Паттерны метапрограммирования.

Основные типы библиотек. Библиотеки функций, процедур и макроопределений. Библиотеки классов. Библиотеки компонентов.

Критерии проектирования стандартных библиотек по составу и функциональности. Библиотеки классов и шаблонов.

Состав и функциональность стандартной библиотеки C++.

Принципы дизайна и реализация стандартной библиотеки шаблонов C++.

Основные виды компонентов STL: контейнеры, итераторы, алгоритмы, аллокаторы.

Тема 9.

30. Реализация основных принципов ООП в программах на языке Фортран.

Инкапсуляция данных с помощью массивов. Перегрузка функций. Производные типы. Классы и объекты. Наследование. Производные типы с указателями, реализация виртуальности.

5. ИСПОЛЬЗУЕМЫЕ ОБРАЗОВАТЕЛЬНЫЕ, НАУЧНО-ИССЛЕДОВАТЕЛЬСКИЕ И НАУЧНО-ПРОИЗВОДСТВЕННЫЕ ТЕХНОЛОГИИ

А. Образовательные технологии.

Работа в аудитории: лекции; консультации перед экзаменом.
Процесс изложения учебного материала сопровождается презентациями. При чтении лекций используются элементы интерактивности – наиболее важные элементы лекций обсуждаются с аудиторией в режиме «вопрос-ответ».

Внеаудиторная работа: изучение пройденных на лекциях тем; самостоятельное изучение литературы по объектно-ориентированным языкам С++ и Фортран; автоматическое тестирование по пройденным темам на платформе TestPad для самопроверки знаний; самостоятельная работа, включающая выполнение практических заданий, предназначенная для закрепления теоретической части курса и получения практических навыков их применения (в объеме 72 часов).

Б. Научно-исследовательские технологии: в ходе самостоятельной работы студенты работают в объектно-ориентированных системах программирования.

В. Научно-производственные технологии: используются технологии работы с современным программным обеспечением.

6. УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ. ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ, ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО ИТОГАМ ОСВОЕНИЯ ДИСЦИПЛИНЫ.

А. Учебно-методические рекомендации для обеспечения самостоятельной работы студентов: в ходе самостоятельной работы студенту следует использовать конспект лекций, основную и дополнительную литературу по курсу и Интернет-ресурсы.

Б. Примерный список заданий для проведения текущей и промежуточной аттестации (темы для докладов, рефератов, презентаций):

Вариант письменной контрольной работы (образец)

1. Вычеркните только те строки функции *main ()*, которые приводят к синтаксической ошибке (если таковые имеются). Обязательно **объясните** причины ошибок. Что будет напечатано в результате работы получившейся программы.

```
struct T {
        int i;
        T(){i = 3;}
        T(int t){ i = t;}
        T operator*(const T & a){
                T t;
                t.i = i * a.i;
                return t;    }
};
struct P:T {
        P operator*(const P & a){
                P t;
                t.i = i * a.i * 2;
                return t;  }
};
```

```
int main () {

        T a(5);
        P b;
        a = a * b;
        cout << a.i << '\n';
        b = b * a;
        cout << b.i << '\n';
        return 0;
}
```

2. Исправьте **только описание класса A** так, чтобы в приведенной ниже программе не было ошибок, а на экран напечаталось **f (A, A)**.

```
struct A {  int n;
            A (int  m) { n = m; }
            operator int () { return 1; }
};
void  f ( int i,  int  j )  { cout << "f(int,  int)\n";}
void  f ( A b,  A a )  { cout << "f(A,  A)\n";}

int main () {       A a (1);
                    f (a, 1);
                    return 0;   }
```

3. Опишите необходимые классы так, чтобы в заданной функции *main ()* не было ошибок, а на экран печаталось **520.**

```
 int main () {
        C c (5);
        B <C> b1(&c), b2(0);
        cout  <<  (*b1). n  <<  (*b2). n  <<  endl;
        return 0 ; }
```

4. Добавьте в описание класса T **из задачи 1** метод

virtual T& **operator --** (){ i = i - 1; cout << i <<" T::**operator --** () \n"; **return *this;** } ,

а в описание (производного от класса T) класса P из задачи 1 -

P& **operator --** () { cout << i << " P::**operator --** () \n"; **return *this;** }

Что напечатает программа с нижеприведенной функцией *main()* и получившимися классами T и P?

```
int main () {
        P b;
        T  a(5),  & rt = b;
        --a;
        --rt;
        rt = a;
        --rt;    return 0;
}
```

5. Опишите на языке Си функцию, проверяющую, является ли заданная строка палиндромом.

6. Укажите все прототипы конструкторов и деструкторов в порядке их выполнения в следующей программе:

```
class A {};
class B: public A{};
struct T { B * pb; };
```

```
int main () {
      A a;
      { a = B();
        T t; }
      B b1, b2 = b1;
      return 0; }
```

7. Добавьте в следующую программу необходимые описания так, чтобы в ней не было ошибок.

```
int main () {
        B::y = 3 ;
        const B b;
        cout << b.x-B::h(2)<< b.h(0)<< endl;
        return 0; }
```

8. Модифицируйте функцию *main(),* **ничего в ней не удаляя, не используя комментарии, goto и прочие переходы** так, чтобы программа завершалась нормально (не аварийно).

```
struct B { virtual void  g() { cout << "B::g () \n"; } };
struct D : B { };

int main () {
        D d, * pd1, *pd2;
        B b, * pb = &b, * pbd = &d;
        pd1 = dynamic_cast < D* > (pbd);
        pd2 = dynamic_cast < D* > (pb);
        if ( typeid (*pd1) == typeid (*pd2) )    pb -> g () ;
        return 0; }
```

9. Вычеркните неверные конструкции в следующей программе на C++11.
Обязательно **объясните** причины ошибок в вычеркнутых конструкциях.

```
struct S {
        int  x = 1;
};

void  g (S && s) { cout << s.x << endl;}

int main () {
        int && i;
        auto k = nullptr;
        decltype ( S ( ) ) a;
        char * s = k;
        k = s;
        g (a);
        g ( S() );
        return 0;
}
```

10. **STL.** Напишите шаблонную функцию, подсчитывающую сумму первых семи элементов заданного контейнера (списка или вектора, константного или неконстантного). Тип элементов контейнера является числовым типом. Если в заданном контейнере меньше 7 элементов, функция должна вернуть 0 соответствующего типа.

В. Примерный список вопросов для проведения текущей и промежуточной аттестации.

1. Абстрактные типы данных, инкапсуляция, наследование, полиморфизм.
2. Класс, объект, состояние объекта, поведение объекта. Диаграммы UML.
3. C++: Пространства имен. Пространство имен *std*.
4. C++: Конструкторы и деструкторы.

5. C++: Присваивание и инициализация.
6. C++: Ссылки в C++. Передача параметров по ссылке.
7. C++: Манипуляции с состоянием объекта.
8. C++: Работа с динамической памятью.
9. C++: Друзья класса.
10. C++: Статические члены класса.
11. Виды полиморфизма в C++ (статический, динамический, параметрический).
12. C++: Статический полиморфизм. Перегрузка бинарных операций:
    а) с помощью функции-члена класса
    б) с помощью функции-друга класса
13. C++: Статический полиморфизм. Перегрузка унарных операций:
    а) с помощью функции-члена класса
    б) с помощью функции-друга класса
14. C++: Специфика перегрузки операций инкремента и декремента, операции индексации.
15. C++: Статический полиморфизм. Перегрузка функций.
16. C++: Алгоритм поиска оптимально отождествляемой (best-matching) функции.
17. C++: Средства обработки ошибок. Исключения и обработка исключений.
18. Виды отношений между классами (ассоциация, наследование, агрегация, использование).
19. C++: Одиночное наследование. Правила наследования. Видимость при наследовании.
20. C++: Динамический полиморфизм. Виртуальные функции.
21. Принципы реализации виртуальных функций
22. C++: Абстрактные классы.
23. C++: Множественное наследование. Видимость при множественном наследовании. Виртуальные базовые классы.
24. C++: Динамическая информация о типе (RTTI).
25. C++: Параметрический полиморфизм. Шаблонные функции.
26. C++: Шаблонные классы.
27. Стандартная библиотека C++.
28. Стандартная библиотека шаблонов STL.
29. STL: контейнеры, итераторы, алгоритмы, аллокаторы.
30. STL: Шаблонные классы *vector* и *list*.
31. Фортран: инкапсуляция, наследование, полимофизм.
32. Фортран: реализация классов.

7. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

7.1. Основная литература:

	Автор	**Название книги/статьи**	**Место издания**	**Изд-во**	**Год издания**
1	Бьёрн С.	Язык программирования C++. Краткий курс	Москва	Бином	2017
2	Буч Г.	Объектно-ориентированный анализ и проектирование с примерами приложений	Москва	Вильямс	2008
3	Шилдт Г.	C++ для начинающих.	СПб	Питер	2024

8. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

А. Помещения: Оборудованные лекционные аудитории.

Б. Оборудование: Ноутбук, мультимедийный проектор, экран для демонстрации решения задач в интерактивном режиме, компьютерный класс для практической апробации студентами лекционных примеров.

Автор-составитель: к.ф.-м.н., доцент факультета ВМК МГУ Вылиток А.А.