ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И КИБЕРНЕТИКИ

Кафедра Алгоритмических языков

(название кафедры)

Согласовано:
Заместитель декана
_____
Факультета ВМК МГУ

Утверждено:
на заседании кафедры
протокол № _____ от
_____

Зав. кафедрой
_____

Учебно-методический комплекс

Дисциплины **«Объектно-ориентированное программирование»**

(наименование дисциплины)

_____

(Цикл дисциплины и его часть (базовая, вариативная, дисциплина по выбору))

Направление подготовки

010300_«Фундаментальные информатика и информационные технологии»

_____

(наименование ООП ВПО направления(й) подготовки, или специальности с указанием кода)

Профиль подготовки

Бакалавр

_____

Разработчик (составитель) Подпись
УМК
К.ф.-м.н. Кузина Л.Н. «___»_____20__ Г.
К.ф.-м.н. Полякова И.Н.

Москва 2013

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное образовательное учреждение
высшего профессионального образования
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет вычислительной математики и информатики

УТВЕРЖДАЮ

Декан факультета ВМК

_____Е.И. Моисеев

«_____»_____2013

Учебно-методический комплекс

«Объектно-ориентированное программирование»

Направление подготовки
010300 «Фундаментальные информатика и информационные технологии»
Бакалавр

Квалификация (степень) выпускника
Бакалавр

Форма обучения
очная

Москва

2013

Рабочая программа дисциплины «Объектно-ориентированное программирование» / составители к.ф.м.н. Кузина Л.Н., к.ф.м.н. Полякова И.Н.

Рабочая программа предназначена для преподавания дисциплины «Объектно-ориентированное программирование» базовой части ЕН цикла студентам очной формы обучения по направлению подготовки «*010300* Фундаментальные информатика и информационные технологии» в 3 семестре.

Рабочая программа составлена с учетом Федерального государственного образовательного стандарта высшего профессионального образования по направлению подготовки, утвержденного приказом Министерства образования и науки Российской Федерации от "8" декабря 2009 г. № 712, а также образовательного стандарта МГУ бакалавр по направлению «010300 Фундаментальные информатика и информационные технологии».

Составители:
к.ф.м.н. Кузина Л.Н.,

к.ф.м.н. Полякова И.Н.

Содержание

1. Цели и задачи освоения дисциплины

Целью освоения дисциплины 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:

1) изучить основные принципы объектно-ориентированной парадигмы программирования, как наиболее распространенной и востребованной в настоящее время;

2) изучить основные возможности объектно-ориентированного языка программирования C++;

3) изучить основные методы программирования на языке C++;

4) получить навыки практического программирования на языке C++.

2. *Место дисциплины в структуре ООП бакалавриата*

Дисциплина относится к базовой части профессионального цикла. Содержание курса определяется образовательным стандартом МГУ высшего профессионального образования по направлению 010300 Фундаментальные информатика и информационные технологии.

Изучаются объектно-ориентированный подход к программированию, объектно-ориентированный язык программирования 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.

Освоение данной дисциплины (модуля) необходимо студентам (независимо от их дальнейшей специализации) для получения навыков объектно-ориентированного анализа и проектирования, а также программирования в объектно-ориентированном стиле, весьма актуальном на сегодняшний день.

3. Требования к результатам освоения содержания дисциплины

Процесс изучения дисциплины направлен на формирование элементов следующих компетенций в соответствии с ФГОС ВПО по данному направлению:

а) **общекультурных (ОК):** владеть основными методами, способами и средствами получения, хранения, переработки информации, иметь навыки работы с компьютером как средством управления информацией;

б) **профессиональных (ПК):**

Способность понимать и применять в исследовательской и прикладной деятельности методы объектно-ориентированного программирования, в том числе:

- способность понимать концепции и использовать на практике следующие базовые знания:

➢ ***на уровне технической грамотности:*** постулаты объектно-ориентированного программирования;

➢ ***на уровне понимания концепций, способности их использования:***
- абстрактный тип данных,
- классы и объекты,
- полиморфизм (динамический – механизм виртуальных функций, статический – перегрузка функций и операций)
- наследование;
- аппарат исключений,

- способность понимать и самостоятельно разрабатывать алгоритмы для решения задач системного и прикладного программирования, исследования математических, информационных и имитационных моделей по тематике выполняемых работ;
- владение основами объектно-ориентированного языка C++;
- понимание концепций и способность использовать в профессиональной деятельности основные законы естественнонаучных дисциплин*;*

в) в результате освоения дисциплины студент должен:

знать основные понятия и концепции объектно-ориентированной парадигмы;

уметь применять на практике основные методы объектно-ориентированной парадигмы;

понимать и применять на практике компьютерные технологии для решения различных задач в объектно-ориентированном стиле;

уметь

- находить, анализировать и контекстно обрабатывать научно-техническую информацию;
- извлекать полезную научно-техническую информацию из электронных библиотек, реферативных журналов;
- демонстрировать способность к анализу и синтезу;
- демонстрировать способность к письменной и устной коммуникации на русском языке;
- публично представить собственные и известные научные результаты;
- очно представлять математические знания в устной форме;

владеть

- навыками решения практических задач объектно-ориентированного программирования;
- методами объектно-ориентированного программирования;
- проблемно-задачной формой представления естественнонаучных знаний.

4. Структура дисциплины (модуля) и ее место в учебном плане

4.1 Тематический план курса

№	Название темы	Аудиторные занятия (часы)		Самостоятельная работа студента
		лекции	семинары	
1.	Принципы объектно-ориентированного программирования.	2	2	12
2.	Определение и использование абстрактного типа данных. Обработка исключений.	6	6	20
3.	Перегрузка функций и операций.	4	4	16
4.	Наследование.	6	6	24
	Итого:	18	18	72
	Всего (аудиторные занятия и самостоятельная работа):	108		

Общая трудоемкость дисциплины составляет 3 зачетных единицы (108 часов).
Лекции – 18 часов, семинары – 18 часов, самостоятельная работа - 72 часа.

4.2. Структура дисциплины по видам работ

№ п/п	Раздел Дисциплины	Неделя семестра	Виды учебной работы, включая самостоятельную работу студентов и трудоемкость (в часах)			Формы текущего контроля успеваемости *(по неделям семестра)* Форма промежуточной аттестации *(по семестрам)*
			лекц	прак	сам	
1	Принципы объектно-ориентированного программирования.	1-2	2	2	12	Самостоятельная работа, индивидуальный опрос
2	Определение и использование абстрактного типа	3-8	6	6	20	Самостоятельная работа, индивидуальный

	данных. Обработка исключений.					**опрос**
3	**Перегрузка функций и операций.**	9-12	4	4	16	**Самостоятельная работа, индивидуальный опрос**
4.	**Наследование.**	13-18	6	6	24	**Самостоятельная работа, индивидуальный опрос, прием практического домашнего задания (16 неделя) Контрольная работа. Коллоквиум (18 неделя) экзамен**

4.3. Лабораторные работы

Лабораторные работы не предусмотрены учебным планом.

4.4. Курсовой проект (курсовая работа, расчетно-графическое задание, реферат, контрольная работа)

Курсовая работа не предусмотрена учебным планом

4.5. Консультации

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

4.6. Интерактивные образовательные технологии, используемые в аудиторных занятиях

Используются традиционные технологии проведения лекций и практических занятий в аудиториях, а также чтение лекций с использованием слайдов. Все методические материалы для прохождения дисциплины отражены на сайте в Интернете.

5. Содержание дисциплины «Объектно-ориентированное программирование»

5.1. Содержание лекций

Лекция №1

Объектно-ориентированное программирование (ООП) – новая технология (парадигма) программирования. Процессно-ориентированный и объектно-ориентированный подходы к программированию. Основные свойства языка, поддерживающего ООП: абстракция, инкапсуляция, наследование, полиморфизм. Объектно-ориентированный анализ и объектно-ориентированное проектирование. Понятие объекта. Выделение используемых объектов, фиксация связей между объектами, фиксация методов обмена сообщениями между объектами.

Лекции №2-3

История возникновения и развития языка C++. Язык C++ в сравнении с языком Си. Обзор средств языка C++, реализующих механизмы ООП. Пространства имен, разрешение области видимости. Понятие класса в языке C++. Описание класса: члены-данные и члены-функции. Управление доступом к членам класса – public, private.
Специальные функции – конструкторы и деструктор. Перегрузка конструкторов. Конструктор умолчания. Конструктор преобразования.
Понятие ссылки на объект. Передача параметров в функции по ссылке. Возврат результата из функции по ссылке.
Конструктор копирования, генерация конструктора копирования по умолчанию, определение конструктора копирования. Указатель this.
Конструктор копирования и операция присваивания: содержательная связь и различие.
Использование квалификатора const в C++. Квалификатор mutable.

Лекция №4

Статические члены класса.
Функции - друзья класса. «Законы» дружбы.
Общая схема обработки исключений: try - throw – catch. Правило выбора обработчика исключения. Передача исключения в объемлющий блок (throw;).

Лекция №5-6

Перегрузка функций. Аргументы со значениями по умолчанию. Перегрузка и неоднозначность.
Перегрузка унарных операций с помощью функции-члена класса и с помощью функции-друга класса. Перегрузка бинарных операторов с помощью функции-члена класса и с помощью функции-друга класса.
Особенности перегрузки операций присваивания, индексирования, инкремента, декремента, операции приведения к типу.

Лекция №7-8

Единичное "открытое" (с квалификатором public). Спецификатор доступа к членам класса protected. Функции-члены производного класса - наследование и замещение, правила видимости. Повторное использование кода.
Конструкторы и деструкторы производных классов: порядок вызова, передача параметров. Указатели на базовый и производный классы, преобразование указателей. Виртуальные функции. Абстрактные классы. Пример использования абстрактного класса. Использование виртуальных деструкторов.

Закрытое и защищенное (private, protected) наследование.

Лекция №9
Коллоквиум.

5.2. Практика. План семинарских занятий.

Номера задач с префиксом «К» даются по задачнику [3] основной литературы (Ю.С. Корухова. Сборник задач и упражнений по языку Си++), с префиксом «Д» – из предлагаемого ниже списка дополнительных задач.

Семинар № 1. «Принципы объектно-ориентированного программирования**»**

Задачи: разбор примеров из лекций.
Выдача контрольного домашнего задания (сдача в конце семестра).
Домашнее задание**:** Д-1

Семинар № 2. «Определение и использование абстрактного типа данных»

Задачи: К-1.1, К-1.4(1,2), К-1.6
Домашнее задание**:** К-1.2, К-1.3, К-1.4(6,7), К-1.5, Д-2

Семинар № 3. Практическое занятие в компьютерном классе.

Задачи: К-1.9.
Домашнее задание**:** К-1.8, К-1.13, Д-3, Д-4

Семинар № 4. «Обработка исключений»

Задачи: К-1.10, К-1.11, К-1.14(б), К-4.5(а), Д-5
Домашнее задание**:** К-1.7, К-1.14(а), К-4.5(б)

Семинар № 5. «Перегрузка функций и операций**»**

Задачи: К-2.1, К-2.4, К-2.7, К-2.8
Домашнее задание**:** К-2.2, К-2.11

Семинар № 6. **Практическое занятие в компьютерном классе**.

Задачи: К-2.6
Домашнее задание**:** Д-6, Д-7

Семинар № 7. «Наследование»

Задачи: К-3.1, К-3.4, К-3.6, Д-8
Домашнее задание**:** К-3.2, К-3.5

Семинар № 8.

Прием контрольного домашнего задания в компьютерном классе (см. 6.3), практическое задание (К-3.3).

Семинар № 9.

Контрольная работа, выполняется на компьютерах (см.6.1)

5.3. Список дополнительных задач

Задача №1.

Выявить в заданной предметной области (ПО) некоторое количество сущностей (понятий). Для каждой из сущностей определить ее структуру, особенности создания, копирования и уничтожения; ее возможное поведение и использование, 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.

Задача №2.

Есть ли ошибки в приведенном фрагменте программы на C++? 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;  }
                void  operator = (X &  a) {i = a.i;     t = a.t;     cout << 3; }
                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 №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 (3);  return y; }
int main ( ) {
        S s (1);
        f (s);
        printf ("%d   ",  s.x);
        return 0;
}
```

Задача №4.

Добавить (если нужно) в класс Cl служебное слово «**const**» так, чтобы заданный фрагмент программы был верным.

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

```
        const Cl   f ( Cl & c) const { cout << c. i << endl;    return *this; }
    };
    const Cl  t1 (const Cl a) {
        Cl b  = Cl (5);
        return b.f ( a );
    }
```

Задача №5.

Объясните, где ошибка в описании класса A? Приведите 2 варианта возможных исправлений приведенной программы, чтобы она стала верной (не вводя дополнительных членов класса)?

```
class A {       int a;
    public:
        static void f (int x) {a = x; }
};
int main () {   A::f (1);       return 0; }
```

Задача №6.

Для класса vect опишите перегруженную операцию индексации так, чтобы были верными все действия функции g (). Возможно ли это? Если нет – **объясните** почему; если можно – опишите и **объясните,** за счет чего удалось добиться требуемого.

```
class vect {
        int *p;
        int size;
public:  vect (int  n) {  p = new  int  [size = n];
                          for (int i =  0;    i <  size;    i++)    p[i] =  i; }
};
void g () {
        vect x (5);
        int k;
        k  = x [2];
        x [1] = x [2 ]+ 5;
}
```

Задача №7.

Описать класс B таким образом, чтобы все конструкции функции main ( ) были верными, класс B содержал только один явно описанный конструктор, а на экран должно быть выдано **17 11 6**.

Нельзя использовать исключения и любые функции досрочного завершения программы.

```
int main () {
        B b1 (1),  b2 (2, 3), b3 (b1);
        b1 += b2 += b3;
        cout << b1.get() << ' ' << b2.get() <<  ' ' <<  b3.get () << endl;
        return 0;
}
```

Задача №8.

Есть ли ошибки в приведенном фрагменте программы на C++? Если есть, то **объясните,** в чем они заключаются. Ошибочные конструкции вычеркнуть из текста программы. Что будет выдано в стандартный канал вывода при вызове функции gg ()?

```
class X { public:  virtual void f ( int i) {cout  << "f_X-int" << endl;}
                   virtual void f ( ) {cout  << "f_X-void" << endl;}
                   virtual  void g(double x) {cout  << "g_X-double" << endl; }
```

```
                    virtual void h (int i) {cout  << "h_X-int" << endl;}
};
class Y:  public X { public:  void f (int j) {cout << "f_Y-int" << endl;}
                              void g(double y) {cout  << "g_Y-double" << endl;}
                    virtual void h ( int i, double t = 1.0) {cout  << "h_Y-int/double" << endl;}
};
void gg ( ) {  X a;    Y b;    X * p  =  & b;
          p -> f (1);     p -> f ( );     p-> g (1.5);     p -> h (10);     p -> h (10, 0.1);
 }
```

6. Оценочные средства для текущего контроля успеваемости, промежуточной аттестации

6.1. Контрольные работы

КОНТРОЛЬНАЯ РАБОТА (проводится на компьютере).

Вариант 1.

Определить класс fridge и, если необходимо, дополнительные функции, таким образом, чтобы следующая функция main работала в соответствии со своим описанием, приведенным в комментариях.

```
int main(){
        fridge R("blue",7,1,12);  //цвет, длина, ширина, высота
        fridge G( 7);    //холодильник белого цвета 7x7x7
        fridge Res = R++; /* оператор ++ увеличивает на 1 все размеры
        Res – копия R до увеличения размеров*/
        cout<<Res<<R<<G;   //вывод информации о холодильниках
        Res =G+R; /*цвет – любой, каждый размер = сумме размеров
холодильников-слагаемых */
        Res=4+G;   //та же функция сложения, что и в предыдущем операторе
        cout<<Res<<R<<G;   //вывод информации о холодильниках
        Res[1]=Res[2]+10;     //длина холодильника = ширина холодильника+10
        cout<<Res<<R<<++G; //вывод информации о холодильниках
        return 0;
}
```

6.2. Оценочные средства для текущего контроля и промежуточной аттестации самостоятельной работы студентов. Коллоквиум.

Вариант письменного коллоквиума (образец)

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

2. Описать класс А так, чтобы: - все конструкции функции main ( ) были верными,
- явно в классе А можно описать не более одного конструктора,
- на экран выдалось **15 60 7**.

Нельзя использовать исключения и любые функции досрочного завершения программы.

```
int main () {
        A a1 (5),   a2 = 4,  a3;
        a2  *=   a1  *=  3;
        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. Внести добавления в описания заданных методов (**не меняя вывод на экран!**) структур B и D так, чтобы все конструкции main () были правильными, а на печать выдалось **5535324242**.

```
struct B {
    float x;
    B (float a) {   x = a;   cout << 5; }
    ~B ( ) {  cout << 2; }
};
struct D : B {
    D ( ) { cout << 3; }
    ~D ( ) {  cout << 4; }
};
int main ( ) {
    B  * p1 = new B (1),  * p2 = new D[2];
    delete p1;
    delete [ ] p2;
    return 0;
}
```

5. 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 );
}
```

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

7. Что напечатает следующая программа?

```
struct B {
        virtual void f (int n) { cout << "f (int) from B\n"; }
        static int i;
};
struct D:  B {
        virtual void f (char n) { cout << "f (char) from D\n"; }
};
int B::i = 1;
int main ( ) {  D  d;  B  b1,  b2,  *pb = &d;
                pb  -> f ( 'a');
                b1.i += 2;      b2.i += 3;      d.i += 4;
                cout << b1.i << ' ' << b2.i << ' ' << d.i << ' ' <<  B::i << endl;
return 0;
}
```

8. Что напечатает следующая программа?

```
struct S {
        S ( int a) {   try  { if (a > 0)  throw  *this;
                              else
                                    if (a < 0)  throw 0;
                       }
                     catch ( S & ) { cout << "SCatch_S&\n"; }
                     catch (int) {  throw; }
                     cout << "SConstr\n";
        }
        S (const S & a) { cout << "Copy\n"; }
        ~S ( ) { cout << "Destr\n"; }
};
int  main ( ) {          try  {    S   s1(1),     s2(-2);
                          cout << "Main\n";
                }
                catch (S  &) {   cout << "MainCatch_S&\n";   }
                catch  ( ... )  {   cout << "MainCatch_...\n";   }
                return 0;
}
```

6.3. Оценочные средства для текущего контроля и промежуточной аттестации самостоятельной работы студентов. Контрольное домашнее задание (КДЗ).

Пример КДЗ.

Задание №1.

Основные понятия и особенности языка С++.

Постановка задачи.

1. *Анализ и проектирование*.

Выбрать для исследования произвольную предметную область (ПО). Выявить в выбранной ПО некоторое количество сущностей (понятий) – *будущие классы*. Для каждой из сущностей определить особенности создания, копирования и уничтожения – *конструкторы и деструктор*, ее возможное поведение и использование – *функции – методы класса и внешние*, а также связи с другими сущностями данной ПО.

2. *Реализация.*

2.1. Реализовать предложенную модель ПО на языке С++.

Требования к реализации модели предметной области.

Обязательно использование в описании ПО следующих возможностей языка С++:

- Иерархия классов

(возможно с использованием абстрактных классов и множественного наследования);
Количество различных классов в программе – не менее трех.

- Перегрузка функций;
- Виртуальные функции;
- Перегрузка операторов = (присваивания) , << (вывода в стандартный поток) и == (сравнения на равенство) для всех обязательна .
  Остальные операторы также могут быть перегружены при необходимости (в зависимости от рассматриваемой области).
- Статические члены классов (хотя бы один);
- «Друзья» класса;
- Исключения.

2.2. Написать программу, которая демонстрирует работу с описанными Вами классами, используя все предложенные Вами возможности. Использовать в программе классы STL (string, vector и т.п.) запрещено.

3. Предоставить отчет - текст программы с комментариями.

Примечание.

Сдача программы подразумевает умение связно отвечать на вопросы по С++ (в рамках пройденного на лекциях и на семинарах), а также способность вносить в программу дополнения и изменения по требованию преподавателя.

7. Оценочные средства рубежного контроля

7.1. Вопросы к экзамену

1. Объектно-ориентированное программирование (ООП) - новая технология (парадигма) программирования. ООП-анализ.
2. Пространства имен в языке Си++.
3. Основные свойства языка, поддерживающего ООП.
4. Понятие класса и объекта. Описание класса.
5. Управление доступом к членам класса - public, private, protected.
6. Операции **.** и **->**, разрешение области видимости, указатель this.
7. Объявления и описания функций-членов класса; эффект inline.
8. Специальные функции - конструкторы и деструктор.
9. Конструктор копирования.
10. Конструктор копирования и операция присваивания: содержательная связь и различие.
11. Ссылки и указатели в Си++.
12. Операторы new и delete.
13. Друзья класса, "законы" дружбы. Сравнение функции-члена и функции-друга: описание, вызов.
14. Обработка исключений в С++.
15. Перегрузка функций. Перегрузка и неоднозначность.
16. Функции с параметрами по умолчанию.

17. Перегрузка операторов. Перегрузка с помощью функции-члена и функции-друга.
18. Перегрузка бинарных операций в C++.
19. Перегрузка унарных операций в C++.
20. Особенности перегрузки операций ++ и --, операции индексации в C++.
21. Особенности перегрузки операции присваивания.
22. Статические члены класса.
23. Константные функции-члены.
24. Одиночное наследование. Правила наследования. Видимость при наследовании.
25. Конструкторы и деструкторы при наследовании.
26. Указатели на базовый и производный классы, преобразование указателей.
27. Динамический полиморфизм. Виртуальные функции.
28. Чисто виртуальные функции. Абстрактные классы.

8. Учебно-методическое и информационное обеспечение дисциплины (модуля)

а) 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. 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

3. 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

б) дополнительная литература:

1. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++. Второе издание, М., Бином, 1998.

2. Страуструп Б. Язык программирования C++. Специальное изд./Пер. с англ. - М.: "Бином", 2008.

3. Страуструп Б. Программирование: принципы и практика использования C++.: Пер. с англ. – М. ООО «И.Д.Вильямс», 2011.

4. Пол А. "Объектно-ориентированное программирование на Си++" – второе издание, М., Бином, 1999.

5. Шилдт Г.. Самоучитель C++. – СПб, BHV, 2001.

6. Страуструп Б. Дизайн и эволюция C++. Пер. с англ. – М.:ДМК Пресс, 2000.

в) Программное обеспечение и Интернет-ресурсы.

Для проведения практических занятий необходимо наличие компилятора языка C++.

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/

9. Материально-техническое обеспечение дисциплины (модуля)

Наличие литературы в библиотеке, медиапроектор и компьютер для проведения лекций-презентаций.

Программа составлена в соответствии с требованиями ФГОС ВПО, ОС МГУ «Фундаментальные информатика и информационные технологии», с учетом рекомендаций Примерной основной образовательной программы (ПрООП) по направлению 010300 «Фундаментальные информатика и информационные технологии», бакалавриат.