Федеральное государственное бюджетное образовательное учреждение
высшего образования
Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики

УТВЕРЖДАЮ
декан факультета вычислительной
математики и кибернетики

/И.А. Соколов /
«27» сентября 2023г.

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

по дисциплине

Объектно-ориентированное программирование

Уровень высшего образования:

бакалавриат

Направление подготовки / специальность:

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

Направленность (профиль) ОПОП:

Искусственный интеллект и анализ данных

Форма обучения:

очная

Рассмотрен и утвержден
на заседании Ученого совета факультета ВМК
(протокол №7, от 27 сентября 2023 года)

Москва 2023

1. ФОРМЫ И ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ И ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ

В процессе и по завершении изучения дисциплины оценивается формирование у студентов следующих компетенций:

Планируемые результаты обучения по дисциплине (модулю)		
Содержание и код компетенции.	**Индикатор (показатель) достижения компетенции**	**Планируемые результаты обучения по дисциплине, сопряженные с индикаторами достижения компетенций**
ОПК-2. Способен применять компьютерные/суперкомпьютерные методы, современное программное обеспечение, в том числе отечественного происхождения, для решения задач профессиональной деятельности	ОПК-2.1. Выбирает компьютерные/суперкомпьютерные методы для решения задач профессиональной деятельности; ОПК-2.2. Использует современное программное обеспечение, в том числе отечественного происхождения, для решения задач профессиональной деятельности	знать • 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.

1.1. Текущий контроль успеваемости

Текущий контроль успеваемости осуществляется путем оценки результатов выполнения заданий практических (семинарских) занятий, самостоятельной работы, предусмотренных учебным планом и посещения занятий/активность на занятиях.

В качестве оценочных средств текущего контроля успеваемости предусмотрены:

контрольная работа
коллоквиум

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 C++.)

Контрольная работа № 2 (проводится на компьютере).

Вариант 1.

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 2.

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()  {
        tiger T1,T2("Kuzya");
        T1.setname("Murzik").setcolor("light").setweight(200);
        //задаем параметры тигра
        T1.print();   //печать информации о тиграх
        T2.print();
        T2.setname("Tigr");
        if (smless (T2,T1))  T2.print();
        //печать информации о том тигре, который по размеру меньше
            else  T1.print();
        //печатаем количество объектов-тигров
        cout<<tiger::count<<endl;
    return 0;
}
```

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);
}
```

1.2. Промежуточная аттестация

Промежуточная аттестация осуществляется в форме экзамена
В качестве средств, используемых на промежуточной аттестации предусматривается:
Билеты

1.3. Типовые задания 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;
}
```

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 два 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;
 }
```

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.

2. КРИТЕРИИ ОЦЕНКИ ПО ДИСЦИПЛИНЕ

ШКАЛА И КРИТЕРИИ ОЦЕНИВАНИЯ результатов обучения (РО) по дисциплине				
Оценка виды оценочных средств	**2 (не зачтено)**	**3 (зачтено)**	**4 (зачтено)**	**5 (зачтено)**
Знания **(виды оценочных средств: приведены в п. 1.2.)**	Отсутствие знаний	Фрагментарные знания	Общие, но не структурированные знания	Сформированные систематические знания
Умения **(виды оценочных средств: приведены в п. 1.2.)**	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)** **(виды оценочных средств: приведены в п. 1.2..)**	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