Министерство науки и высшего образования Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)

Радиофизический факультет

УТВЕРЖДЕНО:
Декан
А. Г. Коротаев

Оценочные материалы по дисциплине

Программирование на C++ часть 3

по направлению подготовки

03.03.03 Радиофизика

Направленность (профиль) подготовки:
Радиофизика, электроника и информационные системы

Форма обучения
Очная

Квалификация
Бакалавр

Год приема
2025

СОГЛАСОВАНО:
Руководитель ОП
М.Л. Громов

Председатель УМК
А.П. Коханенко

Томск – 2025

1. Компетенции и индикаторы их достижения, проверяемые данными оценочными материалами

Целью освоения дисциплины является формирование следующих компетенций:

ОПК-3 Способен понимать принципы работы современных информационных технологий и использовать их для решения задач профессиональной деятельности..

ПК-1 Способен проанализировать поставленную задачу в области радиофизики и электроники, осуществлять поиск, обобщение и использование научно-технической информации, необходимой для эффективного выполнения профессиональной задачи..

ПК-2 Способен проводить математическое моделированию процессов в приборах и устройствах радиофизики и электроники, владеть современными отечественными и зарубежными пакетами программ при решении при решении профессиональных задач..

Результатами освоения дисциплины являются следующие индикаторы достижения компетенций:

ИОПК 3.1 Использует современные информационные технологии и программные средства при решении задач профессиональной деятельности.

ИОПК 3.2 Соблюдает требования информационной безопасности при использовании современных информационных технологий и программного обеспечения.

ИПК 1.1 Понимает требования, предъявляемые к исследуемому прибору, устройству или системе и ожидаемые результаты их использования.

ИПК 1.2 Эффективно осуществляет поиск теоретических и экспериментальных данных в исследуемой и смежных областях деятельности, необходимых для решения поставленной задачи.

ИПК 1.3 Производит сравнительный анализ вариантов решения задачи, определение рисков, связанных с реализацией различных вариантов.

ИПК 2.1 Понимает принцип действия и модели разрабатываемого радиоэлектронного прибора или устройства.

ИПК 2.2 Применяет в профессиональной деятельности различные численные методы, в том числе реализованные в готовых библиотеках при решении конкретных радиофизических задач.

ИПК 2.3 Владеет современными пакетами программ при решении задач в области радиофизики и радиоэлектроники.

2. Оценочные материалы текущего контроля и критерии оценивания

Элементы текущего контроля:

– контрольные вопросы по дисциплине;

– задания для лабораторных работ;

Текущая аттестация проводится в виде устных опросов и лабораторных работ. Опрос состоит из 3 вопросов.

Контрольные вопросы по дисциплине (ИПК 1.1, ИПК 2.1.)

1. Способы создания новых классов в объектно-ориентированной программе. Создание классов с помощью механизма наследования. Перегрузка методов базового класса в порожденном классе.
2. Виртуальные методы.
3. Стандартная библиотека потокового ввода/вывода. Иерархия, назначение основных классов.
4. Стандартные потоковые операторы ввода и вывода. Операторы ввода/вывода для определенных пользователем типов.
5. Состояния стандартных потоков ввода/вывода. Методы, константы, операторы для чтения состояния потока.
6. Методы стандартных потоков ввода/вывода. Методы чтения из потока, методы записи в поток.

7. Манипуляторы стандартной библиотеки потокового ввода/вывода. Принцип работы манипуляторов. Определение собственных манипуляторов.
8. Классы стандартной библиотеки потокового ввода/вывода для работы с файлами. Иерархия классов. Особенности работы. Методы позиционирования.
9. Структура стандартной библиотеки шаблонов (STL). Классификация стандартных контейнеров и итераторов.
10. Стандартные последовательные контейнеры (понятие, характерные свойства каждого контейнера, отличия от ассоциативных контейнеров, трудоемкость выполнения операций и поддерживаемые итераторы).
11. Доступ к элементам контейнера vector с помощью итераторов.
12. Стандартные ассоциативные контейнеры (понятие, характерные свойства каждого контейнера, отличия от последовательных контейнеров, трудоемкость выполнения операций и поддерживаемые итераторы).
13. Специальные итераторы стандартной библиотеки шаблонов (STL). Итераторы потокового ввода/вывода. Итераторы вставки.
14. Стандартные алгоритмы STL. Понятия функторов, объектов функций.
15. Предикаты, операции, функции привязки стандартной библиотеки шаблонов.
16. Стандартные немодифицирующие алгоритмы.
17. Стандартные модифицирующие алгоритмы.
18. Стандартные алгоритмы сортировки и алгоритмы, работающие с отсортированными интервалами.
19. Тестирование программного обеспечения: эвристическое тестирование, тестирование на основе моделей.

Результаты опроса определяются оценками «зачет», «незачет».

Критерии оценивания:

Оценка «зачет» выставляется если даны правильные ответы на два вопроса.

Оценка «незачет» выставляется если даны неправильные ответы на два или три вопроса.

Лабораторная работа состоит из трёх заданий.

Задания для лабораторных работ (ИОПК 3.1., ИОПК 3.2., ИПК 1.3., ИПК 2.3.)

1. Дан класс «длинный булев вектор»:

```
class BitVector
{
private:
   int size;  //размер длинного вектора (число бит)
   int n;// размер массива data – столько элементов типа unsigned long
         // требуется для хранения size бит
   unsigned long *data;//сам вектор

public:
   BitVector();
   BitVector(int num_bits);
   BitVector(const BitVector &v);
   ~BitVector();
   BitVector& operator=(const BitVector &v);
   bool operator==(const BitVector &v) const;
   friend istream& operator>>(istream &s, BitVector &v);
   friend ostream& operator<<(ostream &s, const BitVector &v);
};
```

Добавить в этот класс метод, осуществляющий подстановку (substitution). Данный метод должен формировать новый длинный булев вектор длины m·r на основе

имеющегося булева вектора длины m·k и таблицы, задающей правило подстановки. В таблице задается соответствие некоторых (не обязательно всех возможных!) векторов длины k векторам длины r. В исходном векторе выделяются группы по k бит (всего таких групп будет m), для каждой группы в таблице находится соответствующая группа из r бит. Новый вектор формируется из m таких групп длины r.

Например:

Таблица соответствия:

$a_1 \ldots a_k$	$b_1 \ldots b_r$
111011	0011
000111	1011
111111	0000

Исходный вектор: 000111 111111
Новый вектор: 1011 0000

2. Дан тот же класс «длинный булев вектор», что и в задании 1.

Добавить в этот класс метод, осуществляющий контроль чётности (parity checking). Данный метод должен формировать булев вектор длины 64 по булеву вектору длины 56, добавляя к каждой группе из 7 бит младший бит, значение которого равно 1, если вес группы из 7 бит нечетен, и 0 в противном случае.

Например:

Исходный вектор: 1111111 0000000 1010101 0101010
Новый вектор: 11111111 00000000 01010101 10101010

3. Дан тот же класс «длинный булев вектор», что и в задании 1.

Добавить в этот класс метод, осуществляющий выборку-перестановку (selection-permutation). По заданному булеву вектору длины 64 требуется построить булев вектор длины 48 по следующим правилам: рассматривая исходный булев вектор как вектор, состоящий из двух частей (L и R) длины 32 бит, сформировать требуемый вектор так:

a. бит с номером i *3 (i = 0,1, 2, …, 15) есть бит с номером i части L;
b. бит с номером i *3+1 есть бит с номером i части R;
c. бит с номером i *3+2 есть бит с номером i+1 части R.

Например:

Исходный вектор:
1110111111111101 1111111111110000 01000000000000000000000000100100
Новый вектор: 1 01 1 00 1 00 0 00 1 00 1 00 1 00 1 00 1 00 1 00 1 00 1 00 1 10 0 01 1 00

4. Дан тот же класс «длинный булев вектор», что и в задании 1.

Добавить в данный класс метод, сбрасывающий заданный бит вектора.

Добавить в данный класс метод, сбрасывающий заданную группу битов вектора.

Добавить в данный класс метод, устанавливающий заданную группу битов вектора.

Добавить в данный класс метод, перегружающий оператор побитовой инверсии (~) вектора.

5. Дан тот же класс «длинный булев вектор», что и в задании 1.

Добавить в этот класс метод, осуществляющий перестановку (`permutation`). Данный метод должен формировать новый длинный булев вектор той же длины на основе имеющегося булева вектора и целочисленного массива, задающего правило перестановки. Размер целочисленного массива совпадает с длиной булева вектора (полем `size`), в `i`-ой позиции массива указан номер бита исходного вектора, значение которого нужно записать в `i`-ый бит формируемого вектора.

Целочисленный массив, задающий правило перестановки:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
12	1	0	3	4	5	15	7	8	9	0	11	12	13	14	15

Например:

Исходный вектор: 0111 1111 0000 1110

Новый вектор: 1101 1101 0010 1110

6. Дан тот же класс «длинный булев вектор», что и в задании 1.

Добавить в данный класс метод, перегружающий оператор побитового сложения по модулю два (^) двух векторов.

Добавить в данный класс метод, перегружающий оператор побитового сложения (|) двух векторов.

Добавить в данный класс метод, перегружающий оператор индексации.

Добавить в данный класс метод, устанавливающий заданный бит вектора.

7. Дан тот же класс «длинный булев вектор», что и в задании 1.

Добавить в данный класс метод, перегружающий оператор побитового умножения (&) двух векторов.

Добавить в данный класс метод, инвертирующий заданный бит вектора.

Добавить в данный класс метод, инвертирующий заданную группу битов вектора.

Добавить в данный класс метод, определяющий вес вектора.

8. Дан текстовый файл с координатами вершин треугольников в пространстве. Каждая строка файла содержит координаты вершин одного треугольника. Координата вершины – это тройка вещественных чисел, разделенных пробелом. Координаты двух смежных вершин треугольника также разделяются пробелом. В первой строке содержится общее число треугольников в файле. Например, файл вида:

```
2
1 1 2 1 2 2 1 2 1
3 2 5 2 4 3 5 1 1
```

содержит координаты двух треугольников, первый из которых образован вершинами с координатами (1, 1, 2) (1, 2, 2) (1, 2, 1), попарно соединенных ребрами.

Необходимо построить иерархию классов, соответствующую иерархии типов треугольников (равнобедренный треугольник, равносторонний треугольник, прямоугольный треугольник). В качестве базового класса определить класс `Triangle` (треугольник) с открытым виртуальным методом `Area`, подсчитывающим площадь треугольника в общем случае. Для каждого класса-потомка перегрузить этот метод для оптимального подсчета площади каждого частного случая треугольника. Написать функцию, которая будет считывать треугольники из файла, определять тип каждого треугольника, создавать объект соответствующего класса и помещать его в массив. Определить функцию, которая будет извлекать элементы из построенного массива, и вычислять площади фигур, заданных извлеченными объектами.

9. Написать класс BigNumber для представления больших целых неотрицательных чисел (неограниченной величины). Определить открытый (public) виртуальный метод Out() для вывода чисел на экран и открытый виртуальный метод In() для ввода числа из строки.

Перегрузить для этого класса оператор умножения. Определить метод возведения в степень.

Вывести из класса BigNumber два класса: BigNumberBin, BigNumberHex, для представления больших чисел в двоичной и шестнадцатеричной системах счисления соответственно. Для этих классов перегрузить методы In() и Out().

Ввести с клавиатуры последовательность чисел в разной системе исчисления. Результат поместить в массив объектов типа BigNumber, отсортировать числа по возрастанию и вывести на экран в тех системах исчисления, в которых они были введены.

10. В текстовом файле хранятся записи трех типов: названия отделов предприятия, имена и фамилии сотрудников предприятия и имена и фамилии управляющего персонала предприятия. Каждая запись занимает отдельную строку. Запись о названии отдела предприятия имеет вид:

```
department <название_отдела>:<номер_кабинета>:<описание_отдела>
```

Запись для сотрудника имеет вид

```
employee <имя_сотрудника>:<фамилия_сотрудника>:<название_отдела>
```

Запись для управляющего персонала

```
manager <имя_менеджера>:<фамилия_менеджера>:<количиство_подчинённых>:<фамилия_сотрудника1>, <фамилия_сотрудника2>, ...
```

Примерное содержимое файла:

```
employee Люся:Скворцова:Бухгалтерия
department Бухгалтерия:13:Входи тихо, проси мало, уходи быстро
employee Жора:Пеньков:Финансовый отдел
manager Гоша:Клюквин:2:Скворцова,Пеньков
department Финансовый отдел:14:Бухгалтеры-экономисты
```

Создать класс Records с открытым (public) чисто виртуальным методом Print(). Из класса Records вывести 3 класса, соответствующих трём вышеперечисленным типам записей. Для каждого из этих классов перегрузить метод Print() так, чтобы он выводил:

- Название отдела, номер кабинета, описание – для записи типа “отдел”.
- Имя, фамилию сотрудника и все данные о его отделе из п.1 – для записи типа “сотрудник”.
- Данные об управляющем персонале, всех его подчинённых и их отделах из пп. 2, 1 – для записи типа “управляющий персонал”.

Написать функцию, которая считывает из файла записи и помещает их в массив объектов типа Records, а затем вызывает метод Print() для каждого элемента массива.

11. Написать программу, которая выводит дерево папок и файлов, начиная с заданной пользователем папки. Создать класс FileItem. Определить в нем открытый (public) чисто виртуальный метод Info(). Вывести из класса FileItem классы Folder (класс, инкапсулирующий свойства папки) и File (инкапсулирует свойства файла). Перегрузить в них метод Info(), так, чтобы он выполнял следующие действия.

- В классе File метод Info() выводит на экран имя файла и его размер.
- В классе Folder метод Info() выводит на экран имя папки, меняет текущую директорию на эту папку, получает список папок и файлов из новой директории и формирует соответствующий массив объектов типа FileItem. Затем список сортируется по именам файлов или папок в лексикографическом порядке и для каждого элемента списка

вызывается метод Info().

12. Создать класс Polynom, инкапсулирующий свойства многочленов n-й степени (n > 1) вида
$a_n x^n + a_{n-1} x^{n-1} + ... + a_2 x^2 + a_1 x + a_0 = 0$. Определить открытый (public) виртуальный метод Roots(), для определения корней многочлена. Вывести из класса Polynom классы Polynom1, Polynom2 для представления многочленов 1й и 2й степеней соответственно. Для этих классов перегрузить метод Roots() для нахождения корней многочлена соответствующей степени.

Ввести с клавиатуры несколько многочленов разных степеней. Результат поместить в массив объектов типа Polynom. Для каждого многочлена из массива вывести на экран его корни (если это возможно).

13. Создать класс Cipher, инкапсулирующий свойства алгоритмов шифрования. Определить в нём два чисто виртуальных метода: один для шифровки Encrypt(), другой для дешифровки Decrypt() текстовых сообщений, а также виртуальный метод GetKey() для считывания ключа из текстовой строки. Из класса Cipher вывести два класса PermutationCipher и ReplaceCipher, реализующие алгоритмы любого шифра замены и любого шифра перестановки соответственно. Реализовать функцию EncryptText с тремя аргументами: 1) указатель на объект типа Cipher, 2) ключ 3) имя файла с текстом. Функция считывает из файла текст, шифрует с помощью переданного объекта и ключа и выводит результат обратно в файл. Реализовать функцию DecryptText с тремя аргументами: 1) указатель на объект типа Cipher, 2) ключ 3) имя файла с криптограммой. Функция считывает из файла криптограмму, дешифрует с помощью переданного объекта и ключа, и выводит результат обратно в файл.

14. Считать из двух файлов последовательности вещественных чисел. Если последовательности разной длины, выровнять длины, удалив из более длинной последовательности элементы из начала. Поместить в третью последовательность квадратный корень из суммы квадратов соответствующих элементов первой и второй последовательности. Отсортировать полученную последовательность по убыванию и вывести ее на экран. При программировании не использовать циклов, глобальных и статических переменных.

15. Считать из файла последовательность целых положительных чисел произвольной длины. Выделить из последовательности числа, которые делятся на 3 и поместить их в конец последовательности, отсортировать обе части получившейся последовательности по возрастанию и вывести ее на экран. При программировании не использовать циклов, глобальных и статических переменных.

16. Дан текстовый файл, состоящий из слов, разделенных пробелом, табуляцией или символом новой строки. Подсчитать частоты встречаемости каждого слова этого файла и вывести их на экран. При программировании не использовать циклов, глобальных и статических переменных.

17. Считать последовательности слов из двух файлов. Вывести на экран только те слова, которые присутствуют в первом файле, но отсутствуют во втором. При выводе отсортировать слова по возрастанию длины. При программировании не использовать циклов, глобальных и статических переменных.

18. Считать из файла последовательность целых чисел. Удалить повторяющиеся элементы, посчитать среднее арифметическое всех простых чисел из получившейся последовательности и вывести результат на экран. При программировании не использовать циклов.

19. Считать из файла последовательность вещественных чисел. Извлечь квадратный корень из неотрицательных чисел этой последовательности; вывести результат и отрицательные числа на экран, предварительно отсортировав их по абсолютным значениям (модулям). При программировании не использовать циклов, глобальных и статических переменных.

20. Считать из файла текст. Ввести с клавиатуры строку. Проверить содержится ли введенная строка в тексте. Проверку производить без учета регистра. При программировании не использовать циклов.

21. Считать из двух файлов последовательности целых чисел. Если последовательности разной длины, выровнять длины, удалив из более длинной последовательности элементы из хвоста. Сравнить последовательности покомпонентно и вывести на экран количество совпадений. При программировании не использовать циклов.

22. Считать из файла последовательность слов. Каждому слову сопоставить целое число, соответствующее позиции этого элемента, на которой он бы стоял в отсортированном массиве. Например: Грейпфрут – 1, Яблоко – 3, Лимон – 2, Апельсин – 0. Вывести на экран полученную числовую последовательность (1, 3, 2, 0). При программировании не использовать циклов.

Текущая аттестация по лабораторным работам проводится в виде отчетов по лабораторной работе. Результаты выполнения лабораторных работ определяются оценками «зачет», «незачет».

3. Оценочные материалы итогового контроля (промежуточной аттестации) и критерии оценивания

Промежуточная аттестация по дисциплине проводится в форме устного зачёта по теоретическому материалу. Каждый билет для устного зачёта состоит из трех теоретических вопросов. Первый вопрос проверяет ИОПК 3.1 и ИОПК 3.2, второй вопрос проверяет ИПК 1.2., ИПК 1.3., третий вопрос проверяет ИПК 2.2., ИПК 2.3.

В качестве дополнительных вопросов на устном зачёте используются контрольные вопросы по дисциплине, проверяющие ИПК 1.1., ИПК 2.1.

Оценка «зачет» по лабораторным работам приравнивается к ответу на дополнительный вопрос на устном зачёте.

Вопросы к зачету по дисциплине

1. Классы. Инкапсуляция. Способы создания новых классов в объектно-ориентированной программе.
2. Наследование. Перегрузка методов базового класса в производном классе. Преобразование типов внутри иерархии классов.
3. Ссылки. Способы передачи параметров в функцию. Виды конструкторов. Инициализаторы.
4. Полиморфизм. Виртуальные методы. Таблица виртуальных методов. Абстрактные классы.
5. Чистый и параметрический полиморфизм. Шаблонные функции. Шаблонные классы.

6. Исключения. Статические поля и методы.

7. Манипуляторы стандартной библиотеки потокового ввода/вывода. Принцип работы манипуляторов. Определение собственных манипуляторов.

8. Классы стандартной библиотеки потокового ввода/вывода для работы с файлами. Иерархия классов. Особенности работы. Методы позиционирования.

9. Состояния стандартных потоков ввода/вывода. Методы, константы, операторы для чтения состояния потока.

10. Стандартная библиотека потокового ввода/вывода. Иерархия, назначение основных классов.

11. Стандартные потоковые операторы ввода и вывода. Операторы ввода/вывода для определенных пользователем типов.

12. Методы стандартных потоков ввода/вывода. Методы чтения из потока, методы записи в поток.

13. Классы стандартной библиотеки потокового ввода/вывода для работы со строками как с потоками.

14. Стандартные последовательные контейнеры (понятие, характерные свойства каждого контейнера, отличия от ассоциативных контейнеров, трудоемкость выполнения операций и поддерживаемые итераторы).

15. Стандартные ассоциативные контейнеры (понятие, характерные свойства каждого контейнера, отличия от последовательных контейнеров, трудоемкость выполнения операций и поддерживаемые итераторы).

16. Предикаты, операции, функции привязки стандартной библиотеки шаблонов.

17. Специальные итераторы стандартной библиотеки шаблонов (STL). Итераторы потокового ввода/вывода. Итераторы вставки.

18. Структура стандартной библиотеки шаблонов (STL). Классификация стандартных контейнеров и итераторов.

19. Понятие функторов, объектов функций.

20. Итераторы вставки. Стандартные алгоритмы STL. Классификация стандартных алгоритмов. Для каждого класса алгоритмов привести в качестве примера 1-2 алгоритма.

21. Предикаты, операции, функции привязки стандартной библиотеки шаблонов.

Критерии оценивания:

Результаты зачета определяются оценками «зачет», «незачет».

Оценка «зачет» выставляется если, даны правильные и развернутые ответы на все вопросы билета.

Оценка «незачет» выставляется если, дан неправильный ответ на один и более вопросов билета и дан неправильный ответ на дополнительный вопрос.

4. Оценочные материалы для проверки остаточных знаний (сформированности компетенций)

Тест

1. Что такое «полиморфизм»? (ИПК 2.1.)
   а) Способность одних объектов наследовать свойства и поведение других объектов;
   б) Различная интерпретация одного и того же действия в зависимости от типа объекта, на который это действие направлено;
   в) Совмещение в одном объекте данных и методов для работы с ними;
   г) Создание объекта как экземпляра нескольких классов одновременно;
   д) Разделение полей данных и методов на `public` и `private`.
2. Каким свойством должен обладать деструктор в полиморфном классе и почему? (ИПК 1.1.)
   а) Он должен быть виртуальным, потому полиморфизм класса определяется виртуальным конструктором, который может работать только в паре с виртуальным

деструктором;
б) Он должен быть статическим для того, чтобы гарантировать, что для любого экземпляра полиморфного класса деструктор будет один и тот же;
в) Он должен быть виртуальным для того чтобы гарантировать, что для каждого экземпляра класса в иерархии полиморфных классов будет вызываться «свой» деструктор;
г) В полиморфном классе вообще не должно быть явного деструктора, т.к. компилятор сам выполняет все необходимые действия по удалению экземпляров полиморфных классов;

3. Чем объявление класса с помощью ключевого слова `class` отличается от объявления класса с помощью ключевого слова `struct`? (ИПК 1.3.)
   а) при объявлении класса с помощью ключевого слова `struct` можно объявлять только поля данных (методы объявлять нельзя);
   б) при объявлении класса с помощью ключевого слова `struct` все поля данных и методы являются `public`, при объявлении класса с помощью ключевого слова `class` все поля данных и методы являются `private`;
   в) при объявлении класса с помощью ключевого слова `struct` все поля данных и методы являются `private`, при объявлении класса с помощью ключевого слова `class` все поля данных и методы являются `public`

4. Какие конструкторы создаются неявно при отсутствии в классе явных конструкторов? (ИПК 1.1.)
   а) Только конструктор копирования;
   б) Конструктор по умолчанию и чисто виртуальный конструктор;
   в) Конструктор по умолчанию и конструктор копирования;
   г) Конструктор по умолчанию, конструктор копирования и чисто виртуальный конструктор;
   д) Только конструктор по умолчанию

5. Виден ли изнутри статического метода класса неявный параметр `*this`? (ИПК 2.2.)
   а) да;
   б) нет.
6. С помощью какого механизма ООП можно упростить создание нового класса, используя уже созданные классы? Отметьте все правильные ответы. (ИОПК 3.1.)
   а) С помощью индукции;
   б) С помощью наследования;
   в) С помощью статических методов.

7. Сколько фактических параметров передается при вызове методу с *k* формальными параметрами? (ИПК 2.1.)
   а) k
   б) k+1
   в) 2k
8. С какой целью функцию внутри некоторого класса объявляют дружественной данному классу? (ИПК 2.3.)
   а) Чтобы разрешить этой функции доступ к `private`-полям класса;
   б) Чтобы разрешить вызывать эту функцию из методов класса;
9. Сколько явных параметров должно быть у метода, который перегружает унарный оператор? (ИПК 1.2.)
   а) 0
   б) 1

в) 2

10. Чем отличаются статические переменные от нестатических? (ИОПК 3.2.)

а) Статические переменные, в отличие от нестатических, всегда создаются при запуске программы и уничтожаются при её завершении;

б) Статические переменные, в отличие от нестатических, не обязательно объявлять перед использованием;

в) Статические переменные, в отличие от нестатических, могут использоваться только в статических методах.

Ключи: 1 б), 2 в), 3 б), 4 д), 5 б), 6 б), 7 б), 8 а), 9 а), 10 а).

Информация о разработчиках

Лапутенко Андрей Владимирович, к.т.н., кафедра информационных технологий в исследовании дискретных структур радиофизического факультета, доцент