

**МИНОБРНАУКИ РОССИИ**  
**Ярославский государственный университет им. П.Г. Демидова**

Кафедра компьютерной безопасности и математических методов обработки информации

УТВЕРЖДАЮ

Декан математического факультета



Нестеров П.Н.

21 мая 2024 г.

**Рабочая программа дисциплины**  
**Основы программирования**

Направление подготовки (специальности)  
01.03.02 Прикладная математика и информатика

Направленность (профиль)  
«Прикладное программирование и информационные технологии»

Форма обучения очная

Программа рассмотрена  
на заседании кафедры  
от 26 апреля 2024 г., протокол № 8

Программа одобрена НМК  
математического факультета  
протокол № 9 от 3 мая 2024 г.

## 1. Цели освоения дисциплины

Дисциплина "Основы программирования" нацелена на подготовку студентов к деятельности, связанной с разработкой программного обеспечения для решения профессиональных задач, и, является одним из основных предметов, способствующих развитию алгоритмической культуры и компьютерной грамотности студентов.

Курс способствует формированию фундаментальной базы, необходимой для успешного освоения как общепрофессиональных, так и специальных дисциплин, изучение которых связано с созданием информационных систем для различных предметных областей, их анализом, внедрением и сопровождением.

**Целью изучения** данной дисциплины является ознакомление студентов с понятием алгоритма, способами и средствами их представления, различными парадигмами программирования, классификацией и эволюцией языков программирования, и современными тенденциями их развития, а также детальное изучение одного из языков высокого уровня (язык C++).

## 2. Место дисциплины в структуре образовательной программы

Данная дисциплина относится к обязательной части образовательной программы. Для успешного усвоения данной дисциплины необходимо, чтобы студент овладел также знаниями, умениями и навыками, которые формируются в процессе изучения дисциплин: «Математический анализ», «Алгебра и геометрия», «Дискретная математика» - знать основные понятия и методы математического анализа, алгебры, геометрии; уметь решать задачи теории пределов функций, дифференцирования, интегрирования и разложения функций в ряды; владеть навыками использования стандартных методов и моделей математического анализа и их применения к решению прикладных задач. Знания и навыки, полученные при изучении дисциплины "Основы программирования", используются учащимися при изучении последующих общепрофессиональных и специальных дисциплин компьютерного цикла, в частности дисциплин "Информатика", "Практикум по объектно-ориентированному программированию", "Языки и методы программирования", а также "Архитектура компьютеров" и "Операционные системы". Знания и практические навыки, полученные в результате освоения дисциплины, используются студентами при разработке курсовых и дипломных работ, в научно-исследовательской работе.

## 3. Планируемые результаты обучения по дисциплине, соотнесенные с планируемыми результатами освоения образовательной программы

Процесс изучения дисциплины направлен на формирование следующих компетенций в соответствии с ФГОС ВО, ООП ВО и приобретения следующих знаний, умений, навыков и (или) опыта деятельности:

Формируемая компетенция (код и формулировка)	Индикатор достижения компетенции (код и формулировка)	Перечень планируемых результатов обучения
<b>Общепрофессиональные компетенции</b>		
<b>ОПК-5</b> Способен разрабатывать алгоритмы и компьютерные	<b>И-ОПК-5.1</b> Знает основы программирования, один или несколько языков программирования	<b>Знать:</b> – общие принципы построения и использования современных языков программирования высокого уровня; – базовые структуры данных.

программы, пригодные для практического применения		<ul style="list-style-type: none"> <li>– основы программирования на языке C++;</li> <li>– принципы разработки программ и отдельных программных модулей.</li> <li>– принципы процедурного и объектно-ориентированного программирования</li> </ul> <b>Владеть:</b> <ul style="list-style-type: none"> <li>– навыками алгоритмического мышления.</li> <li>– методиками современной технологии программирования.</li> </ul>
	<b>И-ОПК-5.2</b> Умеет разрабатывать алгоритмы и компьютерные программы для практического применения	<b>Знать:</b> <ul style="list-style-type: none"> <li>– средства описания алгоритмов;</li> <li>– различные парадигмы программирования.</li> </ul> <b>Уметь:</b> <ul style="list-style-type: none"> <li>– формализовать поставленную задачу;</li> <li>– составлять программы на языке программирования высокого уровня для решения вычислительных задач и задач обработки информации;</li> <li>– проводить оценку сложности алгоритмов;</li> <li>– пользоваться инструментальными средствами для разработки прикладных приложений.</li> </ul> - тестировать и отлаживать программы;
	<b>И-ОПК-5.3</b> Имеет практический опыт разработки алгоритмов и компьютерных программ для практического применения	<b>Знать:</b> <ul style="list-style-type: none"> <li>– современные технологии программирования</li> </ul> <b>Уметь:</b> <ul style="list-style-type: none"> <li>– работать с современными системами программирования, включая объектно-ориентированные;</li> <li>– оформлять программную документацию</li> <li>– оценивать качество готового программного продукта</li> </ul> <b>Владеть навыками:</b> <ul style="list-style-type: none"> <li>– разработки алгоритмов решения типовых профессиональных задач;</li> <li>– использования инструментальных средств для создания, отладки и тестирования программ и отдельных модулей, библиотек;</li> <li>– работы с научно-технической литературой и технической документацией по программному обеспечению.</li> </ul>

#### 4. Объем, структура и содержание дисциплины

Общая трудоемкость дисциплины составляет **5** зачетных единиц, **180** акад. часов.

№ п/п	Темы (разделы) дисциплины, их содержание	Семестр	Виды учебных занятий, включая самостоятельную работу студентов, и их трудоемкость (в академических часах)						Формы текущего контроля успеваемости  Форма промежуточной аттестации (по семестрам)
			Контактная работа					самостоятельная работа	
			лекции	практические	лабораторные	консультации	аттестационные испытания		
1	Вводная лекция. Алгоритмы и их представление	1	2						
2	Языки программирования. Трансляция и выполнение программ	1	2					1	
3	Введение в язык C++. Структура программы, основные типы данных.	1	2					1	
4	Операции, выражения и операторы в языке C++.	1	2					1	
5	Простейшие средства ввода и вывода.	1	1					1	
6	Выбор вариантов в языке C++.	1	1					2	Контест № 1 Условия и циклы
7	Программирование циклических процессов.	1	2			2		2	Контрольная работа № 1 Контест № 2 Циклы и последовательности
8	Указатели. Работа с динамической памятью. Арифметика указателей. Ссылки.	1	2					2	Контест № 3 Массивы. Линейный поиск
9	Одномерные массивы: статические, динамические. Контейнер vector(STL)	1	6					8	Контест № 4 Массивы. Сортировка Контест № 5 Бинарный поиск
10	Функции, способы передачи параметров. Указатель на функцию.	1	2					2	Контест № 6 Массивы. Разное
11	Символьная информация. Способы представления строк. Функции обработки строк.	1	4					4	Контест № 7 Разбор строк
12	Средства потокового ввода и вывода. Файлы.	1	2					4	Контест № 8 Массивы строк
13	Многомерные массивы. Способы представления.	1	4			2		8	Контрольная работа № 2 Контест № 9 Двумерные массивы Контест № 10 Двумерные массивы
	Итого за 1 семестр		32			4		36	

14	Использование структур для определения пользовательских типов данных.	2	2			1		1	
15	Использование библиотеки OpenGL.	2	2			1		6	Контеcт № 11 Геометрические задачи
16	Жадные алгоритмы.	2	2					6	Контеcт № 12 Жадные алгоритмы
17	Динамическое программирование.	2	4					6	Контеcт № 13 Динамическое программирование
18	Динамические структуры данных. Реализация на базе массива и списка.	2	10			1		8	Контеcт № 14 Стеки, очереди, деревья Контрольная работа № 3
19	Основы объектно-ориентированного подхода.	2	10			1		8	Контеcт № 15 Контрольная работа № 4 ООП
20	Статические и динамические библиотеки.	2	2					1	
						2	0,5	33,5	экзамен
	<b>Итого за 2 семестр</b>		32			6	0,5	69,5	
	<b>ИТОГО</b>		64			10	0,5	105,5	

### Содержание разделов дисциплины:

#### **Тема 1. Вводная лекция. Алгоритмы и их представление**

Задачи курса и порядок его изучения. Понятие алгоритма. Сущность алгоритмизации вычислительных процессов. Данные и алгоритмический процесс. Представление алгоритмов. Машина Тьюринга.

#### **Тема 2. Языки программирования. Трансляция и выполнение программ.**

Понятие языка программирования. Классификация языков, история разработки. Общая характеристика языка программирования C++. Процесс обработки программ на компьютере. Интерпретация и компиляция. Процесс отладки.

#### **Тема 3. Введение в язык C++. Структура программы, основные типы данных.**

Введение в язык C++. Пример простой программы на C. Структура простой программы. Алфавит языка C++, идентификаторы. Использование комментариев. Данные в языке C++. Переменные и константы. Основные типы данных, их размер, машинное представление. Операция sizeof.

#### **Тема 4. Операции, выражения и операторы в языке C++.**

Основные арифметические операции, порядок их выполнения. Операции ++ и --. Преобразование типов, операция приведения. Побитовые операции. Операции, выражения и операторы в языке C++. Составной оператор.

#### **Тема 5. Простейшие средства ввода и вывода.**

Простейшие средства ввода-вывода. Потоки cin, cout. Библиотека iomanip для форматированного вывода.

#### **Тема 6. Выбор вариантов в языке C++.**

Выбор вариантов. Оператор if в полной и сокращенной форме. Вложенные условные операторы. Операции отношения. Понятие "истина" в языке C++. Логический тип данных. Логические операции. Множественный выбор: операторы switch и break. Тернарный условный оператор ?..

#### **Тема 7. Программирование циклических процессов.**

Программирование циклических процессов. Операторы циклов while, for, do.

Ключевые слова break, continue. Операция запятая. Вложенные циклы. Вычисление функции заданной бесконечным рядом. Нахождение корня функции на отрезке. Табулирование функции.

**Тема 8. Указатели. Работа с динамической памятью. Арифметика указателей. Ссылки.**

Структура памяти. Стек, куча. Функции динамического распределения и освобождения памяти. Нахождение адреса переменной: операция &. Первое знакомство с указателями. Типы указателей. Описание указателей. Арифметика указателей. Операция косвенной адресации \*. Операторы new и delete.

**Тема 9. Одномерные массивы: статические, динамические. Контейнер vector(STL)**

Статические массивы. Использование указателей для работы с массивами. Задачи обработки элементов массива: ввод, вывод, линейный поиск, сортировка, бинарный поиск. Использование датчика псевдослучайных чисел для формирования массива. Трудоемкость. Контейнер vector как динамический массив, методы работы.

**Тема 10. Функции, способы передачи параметров. Указатель на функцию.**

Функции в языке C++. Объявление и определение. Формальные и фактические параметры. Способы передачи параметров. Возвращение значения функцией. Оператор return. Локальные переменные функций. Использование указателей и ссылок для передачи параметров между функциями. Механизм вызова функции, понятие стека. Структура программы, состоящей из нескольких функций. Рекурсия, ее использование. Указатели на функции. Передача указателей на функцию в качестве параметров. Шаблоны функций. Аргументы функции main().

**Тема 11. Символьная информация. Способы представления строк. Функции обработки строк.**

Тип данных char. Кодировочные таблицы. Функции библиотеки ctype для категоризации и преобразования символов. Способы представления строк в языках программирования. Строки в стиле C. Библиотек cstring для работы с нуль-терминированными строками. Класс string и методы работы с ним.

**Тема 12. Средства потокового ввода и вывода. Файлы.**

Текстовые и двоичные файлы. Потоки fstream, ifstream и ofstream. Порядок работы с файловыми потоками, обработка ошибок.

**Тема 13. Многомерные массивы. Способы представления.**

Массивы и указатели. Указатели на указатели и многомерные массивы. Функции и многомерные массивы. Массивы указателей. Вектор векторов.

**Тема 14. Использование структур для определения пользовательских типов данных.**

Структуры, их описание, инициализация, доступ к элементам. Вложенные структуры.

Указатели на структуры, их описание и использование.

**Тема 15. Использование библиотеки OpenGL.**

Открытая библиотека OpenGL, базовые возможности. Подключение библиотеки, работа с цветом, основные графические примитивы. Основные шаги для построения минимальной программы. Обработка событий таймера, клавиатуры и мыши. Анимация.

**Тема 16. Жадные алгоритмы.**

Понятие жадного алгоритма, применимость, понятие надежного шага. Задачи: покрытие точек отрезками, задача о выборе заявок, задача о непрерывном рюкзаке. Сжатие данных, алгоритм Хаффмана. Очередь с приоритетами, мин-, макс-кучи. Класс priority\_queue в STL.

**Тема 17. Динамическое программирование.**

Нисходящее, восходящее динамическое программирование. Одномерная динамика: наибольшая возрастающая подпоследовательность, восстановление решения. Двумерная динамика: расстояние редактирования, взвешенное расстояние редактирования. Задача о

рюкзаке с повторениями, рюкзак без повторений. Нахождение числа сочетаний. Динамика по профилю.

**Тема 18. Динамические структуры данных. Реализация на базе массива и списка.**

Однонаправленные, двунаправленные, циклические списки, бинарные деревья.

Эффективная работа с памятью при реализации динамических структур данных.

Минимизация фрагментации динамически распределяемой памяти. Односвязный, двусвязный линейные списки, стек, очередь, очередь с приоритетами, дек, бинарные деревья, деревья поиска, AVL-деревья, деревья общего вида.

**Тема 19. Основы объектно-ориентированного подхода.**

Объектно-ориентированное программирование: реализация в языке инкапсуляции. Понятие класса, объекта. Конструкторы. Деструкторы. Описание своего класса и создание объектов. Друзья класса. Роль слова `static`. Методы класса. Модификаторы доступа. Вызов методов у экземпляров. Переопределение методов класса.

**Тема 20. Статические и динамические библиотеки.**

Препроцессор языка C++. Директива `#define`. Макроопределения с параметрами, их отличие от функций. Директива `#include`. Заголовочные файлы. Директивы `#undef`, `#ifdef`, `#ifndef`, `#if`, `#else`, `#endif`. Понятие условной компиляции. Библиотеки динамической компоновки (DLL), статические библиотеки, управляемые сборки. Примеры разработки статической и динамической библиотек. Явная загрузка динамической библиотеки.

**5. Образовательные технологии, в том числе технологии электронного обучения и дистанционные образовательные технологии, используемые при осуществлении образовательного процесса по дисциплине**

В процессе обучения используются следующие образовательные технологии:

**Вводная лекция** – дает первое целостное представление о дисциплине и ориентирует студента в системе изучения данной дисциплины. Студенты знакомятся с назначением и задачами курса, его ролью и местом в системе учебных дисциплин и в системе подготовки в целом. Дается краткий обзор курса, история развития науки и практики, достижения в этой сфере, имена известных ученых, излагаются перспективные направления исследований. На этой лекции высказываются методические и организационные особенности работы в рамках данной дисциплины, а также дается анализ рекомендуемой учебно-методической литературы.

**Академическая лекция с элементами лекции-беседы** – последовательное изложение материала, осуществляемое преимущественно в виде монолога преподавателя. Элементы лекции-беседы обеспечивают контакт преподавателя с аудиторией, что позволяет привлекать внимание студентов к наиболее важным темам дисциплины, активно вовлекать их в учебный процесс, контролировать темп изложения учебного материала в зависимости от уровня его восприятия.

**Консультации** – вид учебных занятий, являющийся одной из форм контроля самостоятельной работы студентов. На консультациях по просьбе студентов рассматриваются наиболее сложные моменты при освоении материала дисциплины, преподаватель отвечает на вопросы студентов, которые возникают у них в процессе самостоятельной работы.

**Контест** - соревнование по программированию, при котором участники отправляют код с решением данных задач на сайт, предназначенный для их автоматического тестирования.

**6. Перечень лицензионного и (или) свободно распространяемого программного обеспечения, используемого при осуществлении образовательного процесса по дисциплине**

В процессе осуществления образовательного процесса по дисциплине используются:

для формирования материалов для текущего контроля успеваемости и проведения промежуточной аттестации, для формирования методических материалов по дисциплине:

- программы Microsoft Office;
- издательская система LaTeX;
- Adobe Acrobat Reader.

## **7. Перечень современных профессиональных баз данных и информационных справочных систем, используемых при осуществлении образовательного процесса по дисциплине (при необходимости)**

В процессе осуществления образовательного процесса по дисциплине используются:

- Автоматизированная библиотечно-информационная система «БУКИ-NEXT» [http://www.lib.uniyar.ac.ru/opac/bk\\_cat\\_find.php](http://www.lib.uniyar.ac.ru/opac/bk_cat_find.php)
- Электронно-библиотечная система «Юрайт» <https://urait.ru>
- Электронно-библиотечная система «Лань» <http://e.lanbook.com/>
- Электронно-библиотечная система «Консультант Студента»: <https://www.studentlibrary.ru/>

## **8. Перечень основной и дополнительной учебной литературы, ресурсов информационно-телекоммуникационной сети «Интернет» (при необходимости), рекомендуемых для освоения дисциплины**

### **а) основная литература**

1. О. В. Власова, Н. П. Федотова, О. П. Якимова Основы программирования: учебное пособие - Ярославль: ЯрГУ, 2019. <http://www.lib.uniyar.ac.ru/edocs/iuni/20190204.pdf>
2. Конова, Е. А. Алгоритмы и программы. Язык C++ : учебное пособие для вузов — Санкт-Петербург: Лань, 2021. <https://reader.lanbook.com/book/176900>

### **б) дополнительная литература**

1. М. В. Огнева, Е. В. Кудрина Программирование на языке C++: практический курс: учебное пособие для вузов — Москва: Издательство Юрайт, 2022. <https://urait.ru/viewer/programmirovaniye-na-yazyke-s-prakticheskiy-kurs-492984>
2. В. В. Подбельский, С. С. Фомин Курс программирования на языке Си : учебник, М., ДМК Пресс, 2018.
3. Страуструп, Б. Язык программирования C++ для профессионалов / Страуструп Б. - Москва : Национальный Открытый Университет "ИНТУИТ", 2016. - Текст : электронный // ЭБС "Консультант студента" : [сайт]. - URL : [https://www.studentlibrary.ru/book/intuit\\_418.html](https://www.studentlibrary.ru/book/intuit_418.html)
4. Кувшинов Д. Р. Основы программирования: учебное пособие для вузов — Москва: Издательство Юрайт, 2022. <https://urait.ru/viewer/osnovy-programmirovaniya-493460>
5. Лафоре Р. Объектно-ориентированное программирование в C++.- СПб.: Питер, 2016.
6. Кнут Д. Искусство программирования ЭВМ. Т.1, Основные алгоритмы — М.: Мир, 1976.
7. Вирт Н. Алгоритмы и структуры данных: Пер. с англ. - М.: Мир, 1989.

### **в) ресурсы сети «Интернет»**

1. <https://docs.microsoft.com/ru-ru/cpp/?view=msvc-170>

## **9. Материально-техническая база, необходимая для осуществления образовательного процесса по дисциплине**



Материально-техническая база, необходимая для осуществления образовательного процесса по дисциплине включает в свой состав специальные помещения:

- учебные аудитории для проведения занятий лекционного типа
- учебные аудитории для проведения групповых и индивидуальных консультаций,
- учебные аудитории для проведения текущего контроля и промежуточной аттестации;
- помещения для самостоятельной работы;
- помещения для хранения и профилактического обслуживания технических средств обучения.

Помещения для самостоятельной работы обучающихся оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа к электронной информационно-образовательной среде ЯрГУ.

**Автор(ы):**

Старший преподаватель кафедры компьютерной  
безопасности и математических методов обработки информации

О.В. Власова

**Приложение № 1 к рабочей программе дисциплины  
«Основы программирования»**

**Фонд оценочных средств  
для проведения текущего контроля успеваемости  
и промежуточной аттестации студентов  
по дисциплине**

**1. Типовые контрольные задания и иные материалы,  
используемые в процессе текущего контроля успеваемости**

***Контрольная работа №1***

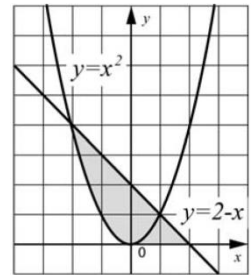
1. Напишите программу, которая вводит с клавиатуры координаты точки на плоскости ( $x$ ,  $y$  – действительные числа) и определяет принадлежность точки заштрихованной области, включая ее границы.

2. Назовём натуральное число  $N$  ( $1000_8 \leq N \leq 777777_8$ ) счастливым, если суммы двух первых и двух последних цифр его восьмеричной записи равны. Найдите количество таких чисел.

Для проверки числа разработайте функцию **bool isHappy(int n);**

3. Ниже записана программа. Получив на вход число  $x$ , эта программа печатает два числа,  $a$  и  $b$ . Укажите наименьшее из таких чисел  $x$ , при вводе которых алгоритм печатает сначала 3, а потом 10.

```
int main()
{
    int x, a, b;
    cin >> x;
    a = 0; b = 1;
    while (x > 0)
    {
        a ++;
        b *= (x % 8);
        x /= 8;
    }
    cout << a << " " << b;
    return 0;
}
```



4. Вводится последовательность целых чисел (0 – конец последовательности), найти разность между наименьшим среди положительных и наибольшим среди отрицательных.

5. Дан целочисленный массив из  $N$  элементов, необходимо найти количество элементов в этом массиве, для которых последняя цифра в шестнадцатеричной записи и в восьмеричной записи одинаковая, и заменить все четные элементы массива на это количество. Гарантируется, что такой элемент есть. Напишите программу для решения этой задачи. В качестве результата программа должна вывести изменённый массив, по одному элементу в строке. Например, для исходного массива из 5 элементов

**22 38 14 23 11**

программа должна вывести числа **3 3 3 23 11**, по одному числу в строке.

В программе разработайте функцию вычисляющую характеристику, функцию изменения массива и функцию печати массива.

### Контрольная работа №2

#### 1. Обработка двумерных массивов

Элементы исходной матрицы вводятся из текстового файла. Результаты выводить на экран и в результирующий текстовый файл. Матрицу выводить до и после преобразований.

Определить сумму и количество положительных чисел расположенных вне диагоналей матрицы  $B(n,n)$ . Если нет положительных чисел, то поменять местами элементы главной и побочной диагоналей.

#### 2. Строки

Описать функцию  $IsIdent(S)$  целого типа, проверяющую, является ли строка  $S$  допустимым идентификатором (набор букв и цифр, начинающийся с буквы). При утвердительном ответе возвращается 0. Если  $S$  является пустой строкой, то возвращается  $(-1)$ , если строка начинается с цифры, то возвращается  $(-2)$ . Если  $S$  содержит недопустимые символы, то возвращается номер первого недопустимого символа. Проверить с помощью этой функции пять данных строк.

### Контрольная работа №3

1. Многочлен  $P(x)=a_nx^n + a_{n-1}x^{n-1} + \dots + a_1x + a_0$  с целыми коэффициентами можно представить в виде линейного односвязного списка, причем если  $a_i=0$ , то соответствующее звено не включать в список. Определить логическую функцию  $bool\ Equal(list\ p, list\ q)$ , проверяющие на равенство многочлены  $p$  и  $q$

2. Даны натуральные  $n$  и  $k$ . По кругу выписывают все натуральные числа от 1 до  $n$ . Сначала отсчитывают  $k$ -ое число, начиная с первого, и удаляют его. Затем от него отсчитывают  $k$  чисел и  $k$ -ое удаляют, и т.д. Процесс останавливается, когда остаётся одно число. Требуется найти это число.

Задача была поставлена Иосифом Флавием (Flavius Josephus) ещё в 1 веке (правда, в несколько более узкой формулировке: при  $k=2$ ).

Решать эту задачу моделированием на базе линейного односвязного списка.

### Контрольная работа №4

1. Написать функцию, подсчитывающую количество вершин в дереве равных  $K$ .  
 $int\ CountVertex(Tree* Top, int K);$

2. Выполнить преобразование дерева. ( $void\ Transform (Tree* Top)$ )

Дан указатель  $Top$  на корень непустого дерева. Для каждой вершины дерева, имеющей две дочерние вершины, поменять местами значения дочерних вершин (т. е. значения их полей  $info$ ).

### Контрольная работа №5

Дано определение класса

*class Matrica*

{ *private: int n,m; int \*\* arr;*

*public :*

*Matrica( int n, int m, int el); //все элементы равны el*

*Matrica( const Matrica &A);*

*Matrica & operator = (const Matrica &A);*

*~ Matrica();*

*friend ostream& operator << (ostream &in, const Matrica &A);*

*Matrica operator+ ( const Matrica &A);*

*int & GetElement(int i, int j);*

*};*

Определите тела для перечисленных методов. Приведите пример программы, демонстрирующий работу с этим классом.

## **2. Список вопросов и (или) заданий для проведения промежуточной аттестации**

1. Алгоритм. Свойства. Способы записи.
2. Машина Тьюринга. Примеры.
3. Архитектура компьютера.
4. Структура программы. Заголовочные файлы, файлы реализации. Препроцессор.
5. Понятие переменной. Типы данных. Области видимости. Время жизни.
6. Арифметические, логические операции. Приоритет.
7. Побитовые операции. Операции сдвига.
8. Арифметика указателей. Связь между указателями и массивами. Ссылки. Примеры.
9. Поточный ввод-вывод (iostream). Использование манипуляторов. Примеры.
10. Условный оператор. Оператор выбора. Примеры.
11. Виды циклов. Примеры.
12. Псевдослучайные числа. Получение целого(вещественного) числа из заданного отрезка [a, b] Примеры.
13. Одномерные массивы. Статическая память. Динамическая память. Примеры.
14. Сортировка выбором. Трудоемкость. Примеры.
15. Сортировка обменом. Модификации. Трудоемкость. Примеры.
16. Сортировка вставками. Модификации. Трудоемкость. Примеры.
17. Быстрая сортировка. Трудоемкость. Примеры.
18. Задачи поиска минимума (максимума) и его номера в одномерном массиве. Примеры.
19. Поиск в одномерном массиве (Линейный, бинарный). Трудоемкость. Примеры.
20. Изменение массива (добавление, удаление элементов). Трудоемкость. Примеры.
21. Статические двумерные массивы. Примеры.
22. Способы создания динамических двумерных массивы. Примеры.
23. Двумерный массив в виде массива массивов. Примеры.
24. Функции. Способы передачи параметров. Примеры.
25. Рекурсивные функции. Алгоритм Евклида (НОД). Примеры.
26. Шаблонные функции. Примеры.
27. Работа с файлами(fstream). Примеры.
28. Функции работы со строками. Библиотека string.h Примеры.
29. Функции работы со строками. Библиотека string Примеры.
30. Структуры (struct). Примеры.
31. Стек. Реализация на базе массива. Примеры.
32. Динамический стек. Реализация на базе линейного списка. Примеры.
33. Стек (STL) Примеры.
34. Очередь. Реализация на базе массива. Примеры.
35. Динамическая очередь. Реализация на базе линейного списка. Примеры.
36. Очередь (STL) Примеры.
37. Очередь с приоритетами. Основные операции. Примеры.
38. Дек. Реализация на базе массива. Примеры.
39. Динамический дек. Примеры.
40. Линейный список. Примеры.
41. Двухнаправленный линейный список. Примеры.
42. Деревья. Способы представления.
43. Деревья общего вида. Реализация. Примеры.

44. Бинарное дерево поиска. Реализация. Обход. Примеры.
45. AVL-дерево. Балансировка. Примеры.
46. AVL-дерево. Алгоритм добавление вершины. Примеры.
47. AVL-дерево. Алгоритм удаления вершины. Примеры.
48. Способы организации дерева произвольного вида. Примеры.
49. Динамическое программирование на примере вычисления чисел Фибоначчи. Примеры.
50. Динамическое программирование сверху вниз. Примеры.
51. Динамическое программирование снизу вверх. Примеры.
52. Динамическое программирование. Нахождение наибольшей возрастающей подпоследовательности. Примеры.
53. Двумерная динамика. Расстояние редактирования. Примеры.
54. Задача о рюкзаке. С повторениями. Примеры.
55. Задача о рюкзаке. Без повторений. Примеры.
56. Жадные алгоритмы. Покрывание точек единичными отрезками. Примеры.
57. Жадные алгоритмы. Задача о выборе заявок. Примеры.
58. Жадные алгоритмы. Задача о непрерывном рюкзаке. Примеры.
59. Жадные алгоритмы. Кодирование Хаффмана. Примеры.
60. Библиотека OpenGL. Структура программы, основные графические примитивы.
61. ООП. Абстракция, инкапсуляция, полиморфизм, наследование.
62. Понятие класса. Поля, методы, перегрузка, this, размеры класса. Примеры.
63. Виды конструкторов. Примеры.
64. Статические поля и методы. Константы в классе. Примеры.
65. Деструкторы. Примеры.
66. Правила перегрузки операций. Перегрузка внешним образом, методами класса. Примеры.
67. Друзья класса. Примеры.
68. Исключения. Примеры.
69. Статическая библиотека.
70. Динамическая библиотека.

### ***Примерный билет по дисциплине***

#### ***Теоретический вопрос***

Задача о рюкзаке. С повторениями. Примеры.

#### ***Задание 1***

Дана строка, заканчивающаяся точкой. Среди символов строки особую роль играет символ #, появление которого в строке означает удаление предыдущего символа. Соответственно, k символов # подряд отменяют k предыдущих символов строки, если таковые имеются. Требуется написать программу, преобразующую строку с учетом указанного значения символа #.

Замечания:

- 1) Если в какой-то момент перед некоторым символом # на этой строке не осталось символов, то его следует игнорировать.
- 2) В выходную строку символы # выводить не следует ни в каком случае.
- 3) Если в результате преобразований все символы в строке были удалены, то следует вывести пустую строку.

Формат входных данных

Признаком окончания ввода служит точка (символ "." ). Строка содержит не более 200 символов.

Входная информация		Выходная информация	
Hello	ww#orld!	#	Hello world
a##abc#			ab
the##he end.			the end.

### Задание 2

Для заданной числовой последовательности  $A[0..N]$  найти длину максимальной подпоследовательности (могут идти не подряд) элементы которой образуют арифметическую прогрессию.

Формат входных данных: Число  $N$  ( $1 \leq N \leq 1000$ ). Элементы последовательности,  $A[i]$  ( $-100 \leq A[i] \leq 100$ ).

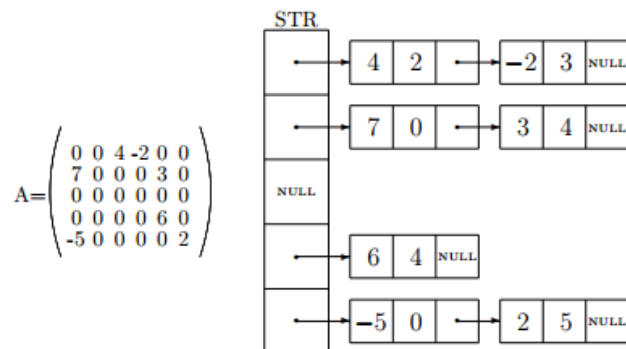
Формат выходных данных: Длина искомой подпоследовательности. Сама подпоследовательность.

Входные данные	Выходные данные
5	3
5 -3 6 15 -10	-3 6 15

### Задание

3

Разреженная квадратная матрица  $A$  (матрица с относительно малым количеством ненулевых элементов) может храниться в памяти в виде набора списков  $STR$ , как показано на рисунке. Каждый ненулевой элемент матрицы хранится в структуре, имеющей три поля: `int Val`; — значение элемента, `int Row`; — номер столбца, `Next` — указатель на следующий ненулевой элемент строки. Компонента  $STR[i]$  массива  $STR$  — это указатель на первый ненулевой элемент  $i$ -й строки. Напишите программу, которая строит разреженную матрицу, затем напишите функцию, которая выполняет преобразование: к элементам главной диагонали прибавьте 2, а затем поменяйте местами строки матрицы (первую с последней, вторую с предпоследней и т.д.).



## **Приложение № 2 к рабочей программе дисциплины «Основы программирования»**

### **Методические указания для студентов по освоению дисциплины**

Основной формой изложения учебного материала по дисциплине «Основы программирования» являются лекции, причем в форме лекции-беседы или мастер-класса. Это связано с тем, что для формирования алгоритмического мышления необходим опыт формализации задач и их поэтапного решения. По большинству тем предусмотрены лабораторные занятия в смежном курсе «Практикум по основам программирования», на которых происходит закрепление лекционного материала путем применения его к конкретным задачам и отработка навыков работы по разработке алгоритмов, кодированию информации или работы с различным программным обеспечением.

Для успешного освоения дисциплины очень важно решение достаточно большого количества задач, требующих разработки алгоритма и написания программы, как в аудитории, так и самостоятельно в качестве домашних заданий. Примеры решения задач разбираются на лекциях и практических занятиях, при необходимости по наиболее трудным темам проводятся дополнительные консультации. Основная цель решения задач – сформировать алгоритмическое мышление. Для решения всех задач необходимо знать и понимать лекционный материал. Поэтому в процессе изучения дисциплины рекомендуется регулярное повторение пройденного лекционного материала. Материал, законспектированный на лекциях, необходимо дома еще раз прорабатывать и при необходимости дополнять информацией, полученной на консультациях, практических занятиях или из учебной литературы.

Большое внимание должно быть уделено выполнению домашней работы. В качестве заданий для самостоятельной работы дома студентам предлагаются задачи, аналогичные разобранным на лекциях и практических занятиях или немного более сложные, которые являются результатом объединения нескольких базовых задач. Задачи сдаются через систему Яндекс-контест, обязательны к решению три задачи из каждого тура.

Для более глубокого усвоения материала в течение обучения проводятся консультации (при необходимости) по разбору заданий для самостоятельной работы, которые вызвали затруднения.

В конце 2 семестра студенты сдают экзамен. Экзамен принимается по экзаменационным билетам, каждый из которых включает в себя один теоретический вопрос и три задачи. На самостоятельную подготовку к экзамену выделяется 3 дня, во время подготовки к экзамену предусмотрена групповая консультация.