ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет автоматики и вычислительной техники

"УТВЕРЖДАЮ"

Декан АВТФ

профессор, д.т.н. Гужов Владимир Иванович

"___"_____ г.

РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ

Информатика

ООП: специальность 230101.65 Вычислительные машины, комплексы, системы и сети

Шифр по учебному плану: ЕН.Ф.3

Факультет: автоматики и вычислительной техники очная форма обучения

Курс: 1, семестр: 1

Лекции: 36

Практические работы: 18 Лабораторные работы: 18

Курсовой проект: - Курсовая работа: - РГЗ: 1

Самостоятельная работа: 70

Экзамен: 1 Зачет: -

Всего: 142

Новосибирск

2011

Рабочая программа составлена на основании _Государственного образовательного стандарта высшего профессионального образования по направлению (специальности): 654600 Информатика и вычислительная техника.(№ 224 тех/дс от 27.03.2000)

ЕН.Ф.3, дисциплины федерального компонента

Рабочая программа обсуждена на заседании кафедры Вычислительной техники протокол № 7 от 31.08.2010

Программу разработал

доцент, к.т.н. Романов Евгений Леонидович

Заведующий кафедрой

профессор, д.т.н. Губарев Василий Васильевич

Ответственный за основную образовательную программу

профессор, д.т.н. Губарев Василий Васильевич

1. Внешние требования

Таблица 1.1

Шифр дисциплины	Содержание учебной дисциплины	Часы
ЕН.Ф.3	Концептуальная записка по направлению подготовки 230100 "Информатика и вычислительная техника" - Информатика: В связи с тем, что предусмотренное стандартом содержание дисциплины "Информатика" частично излагается в дисциплине "Концептуальные основы информатики", а также в связи с необходимостью углубленного изучения программирования как основной составляющей профссиональной подготовки курс в значительной мере соответствует дисциплине "Программирование на языке высокого уровня" стандарта подготовки бакалавров направления 230100 "Информатика и ВТ" (260 часов) - как по содержанию, так и по объему. Вследствие этого дидактические единицы дисциплины и ее содержание формируются из приведенных ниже: ЕН.Ф.02 Информатика: 140 понятие информатики; история развития информатики; место информатики в ряду других фундаментальных наук; мировоззренческие экономические и правовые аспекты информационных технологий; понятие информации и ее измерение; количество и качество информации; единицы измерения информации; информация и энтропия; сообщения и сигналы; кодирование и квантование сигналов; информационный процесс в автоматизированных системах; фазы информационного цикла и их модели; информационный ресурс и его составляющие; информационные технологии; технические и программные средства информационных технологий; основные виды обработки данных; обработка аналоговой и цифровой информации; устройства обработки данных и их характеристики; понятие и свойства алгоритма; принцип программного управления; функциональная и структурная организация компьютера; сетевые технологии обработки данных; виды и характеристики носителей и сигналов; спектры сигналов; модуляция и кодирование; каналы передачи данных и их характеристики; методы повышения помехоустойчивости передачи и приема; современные технические средства обмена данных и	**142**

	каналообразующей аппаратуры; типы и структуры данных; организация данных на устройствах с прямым и последовательным доступом; файлы данных; файловые структуры; носители информации и технические средства для хранения данных; представление информации в цифровых автоматах (ЦА); позиционные системы счисления; методы перевода чисел; форматы представления чисел с плавающей запятой; двоичная арифметика; коды: прямой, обратный, дополнительный, модифицированный; выполнение арифметических операций с числами с фиксированной и плавающей запятой; информационные основы контроля работы цифровых автоматов; систематические коды; контроль по четности, нечетности, по Хеммингу; подготовка, редактирование и оформление текстовой документации, графиков, диаграмм и рисунков; обработка числовых данных в электронных таблицах; основы компьютерной коммуникации. ОПД.Ф.06 Программирование на языке высокого уровня: 260 основные этапы решения задач на ЭВМ; критерии качества программы; жизненный цикл программы; постановка задачи и спецификация программы; способы записи алгоритма; программа на языке высокого уровня; стандартные типы данных; представление основных управляющих структур программирования; теорема структуры и структурное программирование; анализ программ; утверждения о программах; корректность программ; правила вывода для основных структур программирования; инвариантные утверждения; процедуры и функции; массивы; утверждения о массивах; записи; файлы; индуктивные функции на последовательностях (файлах, массивах); динамические структуры данных; линейные списки: основные виды и способы реализации; линейный список как абстрактный тип данных; модульные программы; рекурсивные определения и алгоритмы; программирование рекурсивных алгоритмов; способы конструирования и верификации программ.	

2. Особенности (принципы) построения дисциплины

Таблица 2.1

Особенности (принципы) построения дисциплины

Особенность (принцип)	Содержание
Основания для введения дисциплины в учебный план по направлению или	Рабочая программа составлена на основании Государственного образовательного стандарта высшего профессионального образования (ГОСВПО) по направлению

специальности	230100 (654600) - "Информатика и вычислительная техника". Регистрационный номер и дата утверждения ГОСВПО по направлению 230100 (654600) - "Информатика и вычислительная техника": 224 тех/дс, 27 марта 2000 г. Решение Ученого совета факультета АВТФ протокол №4 от 18.04.2007
Адресат курса	студенты дневной формы обучения по направлению 230100 "Информатика и вычислительная техника" (дипломированные специалисты)
Основная цель (цели) дисциплины	Изучение предметной области информатики в плане представления информации в компьютерах и организации процесса ее обработки в программах.
Ядро дисциплины	Основы представления данных в компьютерах, основы программирования, анализа (понимания) и проектирования алгоритмов и программ на примере "традиционных" задач программирования (арифметические задачи, обработка текста, итерационные циклы, сортировка и поиск)
Связи с другими учебными дисциплинами основной образовательной программы	В связи с тем, что предусмотренное стандартом содержание дисциплины "Информатика" частично излагается в дисциплине "Концептуальные основы информатики", курс частично включает в себя содержание дисциплины "Программирование на языке высокого уровня" стандарта подготовки специалистов по направлению 230100 (654600) - "Информатика и вычислительная техника" - как по содержанию, так и по объему.
Требования к первоначальному уровню подготовки обучающихся	Дисциплина базируется на школьном курсе информатики. Однако практика показывает, что студенты обладают в этом направлении знаниями в основном на уровне представлений, либо фрагментарными сведениями (в основном, учатся "давить кнопки"). Они, как правило, не имеют целостного представления о языке программирования, не имеют понятия о технологии разработки программ (изучают предмет по принципу "вот вам программа, и вот она работает"). Дополнительная сложность - недостаточный уровень математической подготовки: отсутствуют навыки формально-логического мышления при решении сложных задач и составления предварительного плана действий, хотя бы на бумаге. Факт: большинство поступающих не имеют представления о том, что такое процедура (функция) и что такое ее вызов, хотя это основа принципа модульного проектирования программы.
Особенности организации учебного процесса по дисциплине	

3. Цели учебной дисциплины

Таблица 3.1

После изучения дисциплины студент будет

иметь представление	
1	об архитектуре компьютера с точки зрения реализации понятий алгоритма и программы
2	о структуре языков программирования - уровнях описания данных, операций и выражений, операторов, модулей
3	о фазах трансляции и видах ошибок (лексические, синтаксические, семантические, связывания).
4	о средствах определения данных (типы данных, переменные) и алгоритмов, принятых в большинстве языков программирования
5	о технологии проектирования сложных модульных программ
18	о программном представлении моделей физических объектов (геометрия, движение)
знать	
6	формы представления числовой и символьной информации
7	способы определения переменных базовых типов данных и массивов, синтаксис основных операций, операторов и функций в Си
8	структуры данных - последовательность, стек, очередь и способы их представления в массиве
9	назначение ("смысл") переменных в стандартных фрагментах программ, основные принципы анализа программ путем разбиения на стандартные фрагменты
10	основы технологии структурного программирования
11	алгоритмы последовательного и двоичного поиска, виды сортировок, оценки трудоемкости алгоритмов сортировки и поиска
уметь	
12	анализировать существующие и разрабатывать собственные программы с использованием стандартных фрагментов алгоритмов
иметь опыт (владеть)	
13	подготовки, трансляции и отладки программ в среде Microsoft Visual C
14	разработки простых программ на Си и использованием базовых типов данных и массивов
15	использования технологии пошагового модульного проектирования программ
16	проектирования программ обработки символов и строк
17	решения задач, связанных с приближенными вычислениями с использованием итерационных циклов
19	использования графических библиотек для программирования движения простых геометрических объектов
20	разработки функций, имеющих формальные параметры и результат различных типов

4. Содержание и структура учебной дисциплины

Лекционные занятия Таблица 4.1

(Модуль), дидактическая единица, тема	Часы	Ссылки на цели
Семестр: 1		
Модуль: Основы компьютерной архитектуры и языков программирования		
Дидактическая единица: история развития информатики;		
История развития языков программирования	2	1, 2
Дидактическая единица: понятие и свойства алгоритма		
Алгоритмы и программы	2	1, 2
Дидактическая единица: стандартные типы данных		
Язык программирования. Типы данных и переменные	2	1, 2, 3, 4, 6, 7
Дидактическая единица: программа на языке высокого уровня		
Язык программирования. Операции и выражения	2	1, 2, 3, 7
Язык программирования. Фазы трансляции.	2	1, 2, 3
Дидактическая единица: представление основных управляющих структур программирования		
Язык программирования. Операторы.	2	1, 10, 2, 3, 4
Дидактическая единица: процедуры и функции		
Язык программирования. Модульная организация программы	4	1, 10, 2, 3, 4, 5
Модуль: Основы анализа и проектирования программ		
Дидактическая единица: анализ программ		
Основы анализа программ	4	12, 9
Дидактическая единица: основные этапы решения задач на ЭВМ		
Технология разработки программ	4	10, 12, 5, 9
Модуль: "Традиционные" области программирования		
Дидактическая единица: представление основных управляющих структур программирования		
Итерационные циклы	2	10, 17, 5
Обработка текста	4	10, 12, 14, 16, 4, 5, 6, 7
Сортировка и поиск	4	10, 11
Дидактическая единица: критерии качества программы		
Трудоемкость алгоритмов	2	11

Практические занятия Таблица 4.2

(Модуль), дидактическая единица, тема	Учебная деятельность	Часы	Ссылки на цели
Семестр: 1			

Модуль: Основы компьютерной архитектуры и языков программирования			
Дидактическая единица: позиционные системы счисления; методы перевода чисел;			
Системы счисления	Системы счисления (СС), определение числа в позиционной системе счисления, прямой и обратный перевод, оценка значения в 2СС, особенности 8,16СС. Арифметические операции в различных СС. Контрольная работа.	2	1, 6
Дидактическая единица: способы записи алгоритма			
Блок-схемы программ	Общая структура языка. Данные. Переменные. Алгоритм. Блок-схема. Основные управляющие конструкции языка: последовательность, ветвление, цикл, переход. "Интуитивное" программирование - разработка программ с использованием блок-схем. Разработка линейных, разветвляющихся и циклических алгоритмов с использованием блок-схем: арифметические задачи - поиск простых чисел, разложение числа на множители. Контрольная работа.	2	1, 12, 2, 6, 7
Дидактическая единица: представление основных управляющих структур программирования			
Циклические алгоритмы на массивах	Циклические алгоритмы на массивах, "Интуитивное"	2	1, 12, 14, 2, 4, 6, 7, 8

	программирование с использованием блок-схем. Заполнение массива: разложение числа на множители, разложение числа на цифры. Поиск простых чисел (Решето Эратосфена). Контрольное задание.		
Модуль: Основы анализа и проектирования программ			
Дидактическая единица: анализ программ			
Основы анализа программ	"Исторический" и логический анализ программы. "Смысл" выражений и переменных. Примеры построения программ из отдельных смысловых фрагментов - сортировка выбором, вставками, подсчетом. Контрольное задание.	2	12, 2, 6, 9
Дидактическая единица: постановка задачи и спецификация программы			
Проектирование программ	Технология программирования. Примеры проектирования: сортировка погружением, подсчетом, выбором (со сдвигом и обменом) (решение по вариантам, разбор). Образная модель, факты, выстраивание программы методом нисходящего структурного проектирования.	2	10, 12, 5, 9
Модуль: "Традиционные" области программирования			
Дидактическая единица: представление основных управляющих структур программирования			
Символы и строки. Обработка текста	Работа со строками: инвертирование строки,	2	10, 12, 14, 15, 16, 4,

	удаление заданного символа со сдвигом, дублирование заданного символа со сдвигом, подсчет макс. количества повторений (решение по вариантам, разбор). Варианты посимвольной и построчной обработки: инвертирование слов в строке. Контрольное задание.		5, 6, 9
Дидактическая единица: постановка задачи и спецификация программы			
Элементы машинной графики и анимации	Программирование графики. Стандартная графическая библиотека. Элементы машинной графики в прямоугольной и полярной системах координат. Внутреннее представление графического объекта. Моделирование движения: координата, скорость, ускорение. Элементы физического моделирования. Движение тела под действием системы сил. Управление параллллельным перемещением объектов в анимации. Дирсетчер.	4	10, 15, 18, 19, 5
Дидактическая единица: теорема структуры и структурное программирование			
Разработка сложных программ	Модульное проектирование с использованием моделей-заготовок ("грязное" программирование): свертка последовательности повторяющихся символов. Модульное проектирование с	2	10, 12, 14, 15, 4, 5

	использованием функций: сортировка выбором слов в строке по алфавиту, простое однократное слияние. Сложные случаи использования стандартных фрагментов программ. Сортировка однократным слиянием с использованием массива индексов. Циклическое слияние с неявным и явным разделением групп в последовательности		

Лабораторная работа Таблица 4.3

(Модуль), дидактическая единица, тема	**Учебная деятельность**	**Часы**	**Ссылки на цели**
Семестр: 1			
Модуль: Основы компьютерной архитектуры и языков программирования			
Дидактическая единица: основные этапы решения задач на ЭВМ			
Изучение среды программирования MS Visual Studio и анализ простых программ	Изучение оболочки Visual C, приемов редактирования и отладки программ. Стандартные программные контексты, отладка и анализ тестовых заданий	2	12, 13, 14, 3, 4, 9
Модуль: Основы анализа и проектирования программ			
Дидактическая единица: основные этапы решения задач на ЭВМ			
Проектирование программ решения арифметических задач	Разработка программ решения численных задач и задач с массивами (по вариантам).	4	10, 13, 14, 15, 20, 3, 4, 6, 7
Модуль: "Традиционные" области программирования			
Дидактическая единица: основные этапы решения задач на ЭВМ			
Итерационные циклы и приближенные вычисления	Разработать программы приближенного	4	10, 13, 15, 17, 2, 20,

	вычисления суммы ряда (по вариантам). Построить графики полученных зависимостей для эталонной функции и для суммы ряда при заданном числе шагов, а также зависимости числа шагов, необходимых для достижения заданной точности в диапазоне изменения значения X.		3, 6, 7
Работа с символами и со строками	Разработка программы обработки текста (по вариантам)	4	10, 12, 13, 16, 20, 3, 4, 6, 7
Сортировка и поиск	Разработка программы сортировка заданным методом (по вариантам). Получение значений трудоемкости алгоритма путем установки программных счетчиков.	4	10, 13, 14, 19, 20, 3, 4, 5

5. Самостоятельная работа студентов

Семестр- 1, РГЗ
(30 часов)
С использованием простой графической библиотеки в Borland C 3.1 разработать программу с элементами анимации.
Отчет должен содержать описание физической модели, геометрической модели, структурного и функционального описания
программы и описание применения.

Варианты заданий.

1. Программа рисования графиков функций 2 переменных в виде поверхности, образованной линиями, параллельными осям x и y.

2. По экрану движется вращающийся куб, изображаемый в виде граней. Вариант: управление вращением с клавиатуры.

3. По экрану движется вращающийся квадрат, отскакивая от границ экрана, в момент отскока скорость вращения возрастает, с течением времени – уменьшается.

4. Падающий текст. Из текстового файла читаются символы, которые «сыплются» с правого верхнего угла экрана. Начальная скорость варьируется в некоторых пределах. Символы «отскакивают» от нижнего края экрана (неупругое соударение).

5. «Насекомое и лампочка». Насекомое летит на источник света таким образом, чтобы угол между источником и направлением вектора скорости был постоянным, т.е. по спирали. Ударяясь о поверхность лампочки, переходи в режим свободного падения с сохранением горизонтальной составляющей скорости при ударе (отскок от поверхности). Насекомые появляются со случайными начальными точками и скоростями.

6. «Реальный маятник». В маятнике, закрепленном на жесткой оси, составляющая силы тяжести, вызывающая угловое ускорение равна F0*sin(fi), при малых fi?sin(fi) – маятник становится математическим. Изобразить график колебаний маятника y(t).

7. «Грузик» раскачивается и колеблется на пружине. Имеется начальное отклонение грузика от R-R0 – от исходного размера пружины и угол поворота fi. На грузик действуют сила сжатия/растяжения пружины и сила тяжести (проекция m*g*sin(fi) вызывает угловое ускорение m*g*cos(fi) – складывается с силой сжатия/растяжения. Изобразить траекторию движения.

8. «Шарики в силовом поле». Шарики движутся по поверхности экрана, отталкиваясь от стенок. На каждую пару шариков действует сила притяжения вида k/R2-k/R02 , т.е. начиная с R0 притяжение сменяется на отталкивание. (Варианты: 2 шарика, произвольное количество шариков).

9. «Кипящий суп». На дне кастрюли возникают пузырьки, которые всплывают, увеличиваясь в размере. Усложнение: интенсивность появления пузырьков зависит от температуры «дна» T (регулируется) и температуры «воды» (интегрирует разницу T-T0).

10. «Рыбки в аквариуме» плавают, пуская воздушные пузыри.

11. «Светофор».

12. «Шагающий человечек».

13. «Снегопад» - случайным образом падают «снежинки», образуя сугробы по всем поверхностям картинки.

14. «Салют». Движущиеся шарики, увеличивающиеся в размере. Шарики «лопаются», создавая множество мелких, разлетающихся в разные стороны и исчезающих.

15. Анимация сортировок с использованием графической библиотеки. Значение отображается в виде шарика – десятки соответствуют диаметру, единицы – цвету.

16. Анимация обработки строки с использованием графической библиотеки. Символы строки должны изображаться «неровно» (положение, размер) и двигаться неравномерно или с «дрожанием».

Семестр- 1, Подготовка к занятиям

(40 часов)

Для лабораторных занятий - отладка программ, оформление отчетов, выполнение тестовых заданий к защите

Для практических занятий - выполнение контрольных заданий по вариантам, выданным на занятиях

6. Правила аттестации студентов по учебной дисциплине

Для аттестации студентов по дисциплине используется балльно-рейтинговая система. Рейтинг студента по дисциплине определяется как сумма баллов за работу в семестре (текущий рейтинг) и баллов, полученных в результате итоговой аттестации (экзамен, зачет).

В таблице приведено максимальное количество баллов, которое может набрать студент по видам учебной деятельности в течение семестра и диапазоны баллов, соответствующие минимальному и максимальному количествам баллов. Максимальная сумма баллов за семестр составляет 100 баллов (текущий рейтинг - 60 баллов, итоговая аттестация - 40 баллов).

Правила текущей аттестации:

1. В течение первого семестра необходимо представить и защитить 8 лабораторных работ, выполнить контрольные задания (во время практических занятий), расчетно-графическую работу в сроки, установленные учебным графиком.

2. За каждую сданную лабораторную работу и контрольное задание начисляется фиксированное число баллов в зависимости от ее сложности и объема (6 или 10).

3. Снижение исходного балла производится за следующие недочеты:

- существенное вмешательство преподавателя в ход выполнения работы, неумение студента пользоваться методическими материалами, - существенные недостатки и ограничения в выполненной работе, упрощенный вариант задания - до 20%;
- неграмотно или неполно оформленный или неотформатированный отчет - до 10%;
- несвоевременное выполнение работы (определяется сроком предоставления отчета) - по 10% за каждую просроченную неделю, но не более 50%;
- пропуск отведенного под работу занятия по неуважительной причине - 10%.

4. За досрочную сдачу и защиту лабораторных работ студенту начисляются бонусы - по 10% за каждую неделю, но не более 50%;

5. Если при предоставлении отчета по работе студент не ориентируется в учебном материале, не может объяснить сущность работы и ее результаты, либо в отчете не отражены существенные элементы работы, то она возвращается на доработку (с перенесением сроков сдачи на неделю).

6. Защита некоторых лабораторных работ производится на тестовых заданиях (фрагментах программ), для которых требуется выполнить вызов функции и ее анализ (см. http://ermak.cs.nstu.ru/cprog - вопросы без ответов). На каждую защиту начисляется фиксированное число баллов. Снижение исходного числа баллов производится за несвоевременную защиту (по 10% за каждую просроченную неделю, но не более 50%).

7. Срок защиты устанавливается "+2 недели" от срока выполнения.

8. Если на защиту представлен неполный материал, либо студент не ориентируется в нем, то защита назначается повторно.

9. На контрольные задания и РГР распространяются правила расчета баллов, принятые для лабораторных работ (выполнение, оформление, сроки).

Правила итоговой аттестации:

1. К экзамену допускаются студенты, сдавшие лабораторные работы, контрольные задания, РГР, и набравшие не менее 50% (30 баллов) по результатам текущего рейтинга.

2. Экзамен проводится в устном виде, предлагается теоретический вопрос (15 баллов) и задача (15 баллов) и тестовые задания по материалам контрольных заданий и тестов на защиту лаб.работ (10 баллов).

3. На подготовку теоретического вопроса и задачи выделяется время (1 час). Ответ на тестовые задания производится в "реальном времени". При подготовке к ответу студент может пользоваться вспомогательными учебными материалами. Ответ на теоретический вопрос проводится в форме беседы, в которой преподаватель задает любые вопросы, касающиеся этой темы, как в теоретическом, так и в практическом плане. В задаче проверяется умение студента правильно вносить изменения и дополнения, сформулированные преподавателем (изменения не должны выходить за рамки формулировки задачи).

4. Критерии выставления оценки за теоретический вопрос:
- 15 баллов - развернутые ответы на все вопросы по теме, знание основных терминов и определений по другим темам предмета;
- 10 баллов - неполные ответы на вопрос, либо отрицательные ответы на некоторые из них (в пределах 25%), незнание основных терминов и определений по другим темам предмета;
- 5 баллов - неполные ответы на некоторые вопросы темы (в пределах 25%);
- 3 балла - знание основных терминов и определений при отсутствии ответов на вопросы;
- 0 баллов - незнание основных терминов и определений при отсутствии ответов на поставленные вопросы по теме.

5. Критерии выставления оценки за задачи:
- 15 баллов - задача решена, студент в состоянии внести и обосновать любые изменения в программу "в реальном времени";
- 10 баллов - решение задачи с существенными недочетами и пробелами, либо с отсутствием объяснения, как получены отдельные элементы;
- 5 баллов - решение задачи с существенными недочетами и пробелами без объяснений, как они получены;
- 0 баллов - задача не решена.

Таблица 1.

№ п/п	Вид учебной работы (учебной деятельности)	Вып	Защита	Срок (неделя)
	Первый семестр:			
1.	Лабораторная работа №1. Анализ программ.	3	1	6
2.	Лабораторная работа №2. Арифметические задачи.	6	2	10
3.	Лабораторная работа №3. Итерационные циклы	6	2	12
4.	Лабораторная работа №4. Символы. Строки.	6	2	14
5.	Лабораторная работа №5. Сортировка	6	2	16
6.	Контр. задание №1. Системы счисления	2		2
7.	Контр. задание №2. Разработка программ		2	6
8.	Контр. задание №3. Анализ программ	2		10

9. Контр. задание №4. Символы. Строки.	2	14	
10. Расчетно-графическая работа	16		16
Итого по текущему рейтингу:	60		
Экзамен	40		

7. Список литературы

7.1 Основная литература

В печатном виде

1. Романов Е. Л. Практикум по программированию на С++ : [учебное пособие] / Е. Л. Романов ; Новосиб. гос. техн. ун-т. - СПб., 2004. - 426, [1] с. : ил.
2. Хабибуллин И. . Программирование на языке высокого уровня С/С++ : учебное пособие для вузов по направлению 654600 "Информатика и вычислительная техника" / И. Ш. Хабибуллин. - СПб, 2006. - 485 с. : ил. - Рекомендовано УМО.
3. Подбельский В. В. Программирование на языке Си : учебное пособие для вузов по направлениям: "Прикладная математика и информатика", "Информатика и вычислительная техника", специальностям "Прикладная математика", "Вычислительные машины, комплексы, системы и сети управления" / В. В. Подбельский, С. С. Фомин. - М., 2007. - 600 с. : ил., табл. - Рекомендовано МО.
4. Романов Е. Л. Язык Си++ в задачах, вопросах и ответах : [учебное пособие] / Е. Л. Романов. - Новосибирск, 2003. - 426, [1] с. : ил.

В электронном виде

1. Романов Е. Л. Практикум по программированию на С++ : [учебное пособие] / Е. Л. Романов ; Новосиб. гос. техн. ун-т. - СПб., 2004. - 426, [1] с. : ил.

7.2 Дополнительная литература

В печатном виде

1. Подбельский В. В. Практикум по программированию на языке Си : учебное пособие для вузов по направлениям "Прикладная математика и информатика", "Информатика и вычислительная техника" и специальности "Прикладная математика и информатика" / В. В. Подбельский. - М., 2004. - 574, [1] с. + 1 электрон. опт. диск (CD-ROM). - Рекомендовано МО.
2. Климова Л. М. СИ++. Практическое программирование. Решение типовых задач : учебное пособие / Л. М. Климова. - М., 2001. - 587 с. : ил.
3. Крупник А. Изучаем Си / А. Крупник. - СПб., 2001. - 251, [1] с. : ил.

8. Методическое и программное обеспечение

8.1 Методическое обеспечение

В печатном виде

1. Романов Е. Л. Си/Си ++. От дилетанта до профессионала [Электронный ресурс] : электронное учебное пособие : для 1-2 курсов направления 230100 "Информатика и вычислительная техника / Романов Е. Л. - Новосибирск, 2010. - 1 электрон. опт. диск (CD-ROM). - Загл. с этикетки диска.- Рег. свидетельство №18891.

В электронном виде

1. Романов Е. Л. Си/Си ++. От дилетанта до профессионала [Электронный ресурс] : электронное учебное пособие : для 1-2 курсов направления 230100 "Информатика и

вычислительная техника / Романов Е. Л. - Новосибирск, 2010. - 1 электрон. опт. диск (CD-ROM). - Загл. с этикетки диска.- Рег. свидетельство №18891.

8.2 Программное обеспечение

1. Microsoft, eMbedded Visual C++, Интегрированная среда разработки
2. НГТУ, Романов Е. Л. Си/Си ++. От дилетанта до профессионала [Электронный ресурс] : электронное учебное пособие :, Примеры программ, тестовое ПО

9. Контролирующие материалы для аттестации студентов по дисциплине

Лабораторные и практические занятия - см. п.9. Приложения.

Экзаменационные вопросы и задачи

Теоретические вопросы

1. Позиционная система счисления. Представление чисел. Прямое и обратное преобразование из 10СС в СС с другим основанием. Арифметика в СС с другим основанием. Особенности двоичной СС.

2. Единицы представления информации в компьютере: бит, байт, машинное слово. Двоичная и шестнадцатеричная системы счисления и их использование в компьютере. Оценка информационной емкости двоичного числа. Представление целого без знака в машинном слове.

3. Понятие типа данных и переменной. Определение переменных в Си. Базовые типы данных char,int,long как машинные слова. Особенности типа данных char. Знаковая и беззнаковая формы представления в Си. Представление символов. Представление чисел с плавающей запятой.

4. Массивы: особенности работы, инициализация. Массивы как формальные параметры функций.

5. Выражения и операции (обзор и классификация): арифметические, сравнения, логические, машинно-ориентированные, присваивания, адресные, выделения составляющего типа данных. Особенности выполнения операций на Си (совместимость, приоритеты, направление выполнения, действие и результат). Особенности выполнения арифметических операций и операций присваивания. Операция "запятая". Операции сравнения, логические операции.

6. Преобразование базовых типов данных в выражениях: действия, порядок. Явные и неявное преобразования.

7. Выражения и операторы. Роль ";" как ограничителя. Классификации управляющей логики программы - последовательность, условие, цикл, переход. Основные операторы Си: if, while, do-while, for, switch, break, continue, return, goto: классификация, особенности синтаксиса и выполнения.

8. Функции. Формальные и фактические параметры. Способы передачи параметров – по значению и по ссылке. Результат функции. Локальные и глобальные (автоматические и внешние) переменные. Функция как основа модульного программирования.

9. Структура данных как система взаимосвязанных переменных и значений. Стек. Представление стека в массиве. Свойства. Использование стека при вызове функции.

10.Структура данных как система взаимосвязанных переменных и значений. Последовательность. Стек и очередь. Свойства. Представление очереди в массиве.

11.Циклические программы. Виды циклов. Итерационный цикл. Рекуррентные последовательности и итерационные циклы. Программа вычисления корня функции. Программа вычисления суммы ряда.

12.Работа со строками. Представление строки в Си. Строка и массив символов. Поиск в строке. Посимвольная и пословная обработка

13.Преобразования целого числа из внешней формы во внутреннюю и обратно.

14.Сортировка и поиск. Понятие записи и ключа. Линейный и двоичный поиск. Трудоемкость алгоритмов сортировки и поиска.

15.Классификация сортировок: выбор, вставка, обмен, подсчет, разделение, слияние.

16.Сортировки выбором и вставками, вставка погружением. Трудоемкость.

17. Обменная сортировка и ее оптимизации: шейкер-сортировка, сортировка Шелла.

18. Сортировки разделением и слиянием. Однократное слияние, трудоемкость.

19.Рекурсивное разделение, медиана, способы разделения. «Быстрая» сортировка.

20.Циклическое слияние, трудоемкость.

21.Основы анализа программ. “Исторический” и логический анализ программы. “Смысл” выражений и переменных. Стандартные программные контексты.

22.Технология программирования. Цикл проектирования программы. Образная модель. Предварительный сбор фактов. «Историческое», структурное и «грязное» программирование.

23.Структурное программирование – нисходящее, пошаговое, структурное проектирование программы и данных. Последовательность, ветвление и цикл. Модульное программирование.

24.Трансляция и компоновка программы. Функция, модуль (файл), проект. Объектный модуль, компоновщик. Фазы трансляции: препроцессор, лексический, синтаксический и семантический анализ. Примеры лексических, синтаксических и семантических ошибок.

25.Время жизни и область действия переменных и функций. Определения и объявления. Глобальные(внешние), локальные (автоматические) и статические переменные и их свойства. Библиотеки и заголовочные файлы.
Задачи

Задачи оформляются в виде функций с передачей всех входных и выходных данных через формальные параметры и результат. Задачи, работающие со строками, по заданию преподавателя реализовать либо с использованием массивов char[], либо с использованием указателей char* и операций вида *p++. Для функции привести пример вызова из main со статическими входными параметрами.

1. Найти наименьшее общее кратное для всех элементов массива - минимальное число, которое делится на все элементы массива без остатка.

2. Написать функцию проверки, является ли заданное число простым. С ее помощью написать программу поиска простых чисел в диапазоне 1000-2000, две любые части которого - также простые (например, 1997, 1-997,19-97,199-7)

3. Сформировать массив простых чисел в диапазоне от 2 до заданного. Очередное простое число определяется попыткой деления нацело числа на все уже накопленные простые числа.

4. Любая целочисленная денежная сумма n>7 может быть выдана без сдачи “трешками” и “пятерками”. Программа находит эти числа для заданного n.

5. На заданном интервале найти все числа, цифры которых находятся в строго возрастающем порядке (например, 532).

6. Число Армстронга – такое число из k цифр, для которого сумма k-х степеней его цифр равна самому этому числу, например 153=13+53+33. Найти все трехзначные числа Армстронга.

7. Найти все двухзначные (трехзначные) числа, которые совпадают с последними цифрами своих квадратов, например, 252=625, 762=5676.

8. Число называется совершенным, если оно равно сумме всех своих делителей, например, 6=1+2+3, 28=1+2+4+7+14. Найти все совершенные числа в заданном интервале.

9. Найти все числа в заданном диапазоне, которые делятся на любую из своих цифр.

10. Найти числа в диапазоне 100-10000, для которых куб суммы цифр равен значению самого числа (например, 512 - 5+1+2=8).

11. Разложить число на простые множители (например 36=3*3*2*2). Результат – последовательность множителей в массиве, ограниченная 0.

12. Определить в массиве максимальную длину последовательности расположенных подряд возрастающих значений и возвратить индекс ее начала.

13. Подсчитать количество слов в строке.

14. Найти в строке слово минимальной длины и возвратить индекс его начала.

15. Оставить в строке по одному пробелу между словами.

16. Возвратить индекс начала фрагмента, симметричного относительно центрального символа (например, "abcba"), и имеющего максимальную длину. Длину фрагмента возвратить по ссылке.

17. "Перевернуть" в строке все слова. (Например: "Жили были дед и баба" - "илиЖ илиб дед и абаб").

18. Преобразовать целое из внутренней формы во внешнюю в шестнадцатеричной СС.

19. Преобразовать число в шестнадцатеричной СС из внешней формы во внутреннюю.

20. Преобразовать дробную часть переменной типа double во внешнюю форму представления (строку символов).

21. В строке находится символ “точка” и символы-цифры дробной части числа. Преобразовать во внутреннюю форму представления (переменную типа double).

22. Найти в строке два одинаковых фрагмента, не содержащих пробелы и имеющих максимальную длину и возвратить индекс начала первого из них.

23. В строке, содержащей абзац текста, найти концы предложений, обозначенный символом "точка". В следующих за ними словах первую строчную букву заменить на прописную. Между словами количество пробелов может быть любым.

24. Заменить в строке все восьмеричные целые константы на символы с соответствующими кодами, (например, 101 на A).

25. Найти слово, начинающееся с самой младшей латинской буквы и возвратить индекс его начала.

26. Удалить из строки комментарии вида "/* ... */". Игнорировать вложенные комментарии.

27. Заменить в строке все большие латинские буквы на соответствующие им шестнадцатеричные коды (например, A на 0x41, в константе использовать 2 цифры для представления байта).

28. Заменить в строке все целые константы из 1-2 цифр соответствующим повторением следующего за ними символа (например "abc5xacb15y" - " abcxxxxxacbyyyyyyyyyyyyyyy ").

29. Найти в строке и удалить из нее последовательность повторяющихся символов максимальной длины (например, "abcxxxxxacbyyyyyyyyyyyyyyyz" - "abcxxxxxacbz").

30. Найти в строке наиболее часто встречающийся символ и заменить его на пробел.

31. Найти все вхождения подстроки в строке. Строка и подстрока заданы в массивах символов. Результат – массив, заполненный индексами начала подстроки в строке, последовательность ограниченна -1.

32. Найти в строке самую внутреннюю пару скобок и возвратить индекс открывающейся, например “a(bb(c(d)e(f (ggg)h)))(d)”.

33. Сортировка выбором. Выбирается минимальный элемент в массиве и запоминается. Затем удаляется, а все последующие за ним элементы до конца массива сдвигаются на один влево. Сам элемент заносится на освободившуюся последнюю позицию.

34. Сортировка выбором. Выбирается минимальный элемент в массиве и запоминается. Затем удаляется, а все последующие за ним элементы массива сдвигаются на один влево. Сам элемент переносится в новый массив.

35. Сортировка выбором. Выбирается минимальный элемент в массиве и запоминается. Затем на его место записывается «очень большое число», например, максимальное +1. Сам элемент переносится в новый массив.

36. Сортировка выбором. Выбирается максимальный элемент из оставшихся и меняется с последним из них (обратить внимание на диапазон неупорядоченной части).

37. Сортировка выбором. Выбирается минимальный элемент из оставшихся и меняется с первым из них (обратить внимание на диапазон неупорядоченной части).

38. Сортировка подсчетом. Выходной массив заполняется значениями “-1”. Затем для каждого элемента определяется его место в выходном массиве путем подсчета количества элементов строго меньших данного. Естественно, что все одинаковые элементы попадают на одну позицию, за которой следует ряд значений “-1”. После чего оставшиеся в выходном массиве позиции со значением “-1” заполняются копией предыдущего значения.

39. Сортировка вставками. Берется очередной элемент и извлекается из массива. Затем от начала массива ищется первый элемент, больший данного. Все элементы, от найденного до очередного сдвигаются на один вправо и на освободившееся место помещается очередной элемент. (Поиск места включения от начала упорядоченной части).

40. Сортировка вставками («всплытием»). Берется предпоследний элемент и меняется со следующим, пока не закончится массив и пока следующий будет меньше текущего. Затем берется предыдущий и т.д..

41. Сортировка вставками. Берется очередной элемент массива. Затем от начала выходного массива ищется первый элемент, больший данного. Все элементы, от найденного до последнего сдвигаются на один вправо и на освободившееся место помещается очередной элемент.

42. Сортировка выбором. Выбирается минимальный элемент в массиве и переносится в новый массив. Затем на его место записывается последний элемент исходного массива.

Лабораторные работы . Варианты заданий и тесты на защиту

2.1 Арифметические задачи

1. Найти в массиве и вывести значение наиболее часто встречающегося элемента.

2. Найти в массиве элемент, наиболее близкий к среднему арифметическому суммы его элементов.

3 Найти наименьшее общее кратное всех элементов массива (то есть числа, которое делится на все элементы).

4 Найти наибольший общий делитель всех элементов массива (на который они все делятся без остатка).

5 Интервал между минимальным и максимальным значениями элементов массива разбить пополам и относительно этого значения разбить массив на две части (части не сортировать).

6 Задан массив, определить значение k, при котором сумма |A(1)+A(2)+…A(k)-A(k+1)+…+A(N)| минимальна (то есть минимален модуль разности сумм элементов в правой и левой части, на которые массив делится этим k).

7 Заданы два упорядоченных по возрастанию массива. Составить из их значений третий, также упорядоченный по возрастанию (слияние).

8 Известно, что 1 января 1999 г. – пятница. Для любой заданной даты программа должна выводить день недели.

9 Известно, что 1 января 1999 г. – пятница. Программа должна найти все «черные вторники» и «черные пятницы» 1999 года (то есть – 13 числа).

10 Найти в массиве наибольшее число подряд идущих одинаковых элементов (например {1,5,3,6,6,6,6,6,3,4,4,5,5,5} = 5).

11 Составить алгоритм решения ребуса РАДАР=(Р+А+Д)^4 (различные буквы означают различные цифры, старшая - не 0).

12 Составить алгоритм решения ребуса МУХА+МУХА+МУХА=СЛОН (различные буквы означают различные цифры, старшая - не 0).

13 Составить алгоритм решения ребуса ДРУГ-ГУРД=2727 (различные буквы означают различные цифры, старшая - не 0).

14 Составить алгоритм решения ребуса КОТ+КОТ=ТОК (различные буквы означают различные цифры, старшая - не 0).

15. Несократимая дробь задана числителем и знаменателем – переменными типа long. Разработать функцию сложения дробей.

Тестовые задания

Содержательно сформулировать результат выполнения функции, определить «смысл» отдельных переменных, найти стандартные контексты, их определяющие, написать вызов функции.

Пример выполнения тестового задания.

```
//-------------------------------------------------------21-07.cpp
//-------------------------------------------------------
 int test(int a){
         int n,k,i;
 for (n=a; n!=0; n/=10){
```

```
        k=n%10;
    if (k==0) break;                    // Цифра - 0
        if (k==1) continue;               // Цифра - 1
    for (i=2; i<k; i++)
        if ( k%i==0) break;             // Цифра не простая
    if (k!=i) break;                    // Цифра не простая
    }
if (n==0) return 1;                           // Дошли до конца без break -
return 0; }                         // все цифры простые.

#include <stdio.h>
void main(){
printf("test(1357)=%d\n",test(1357));
printf("test(1457)=%d\n",test(1457)); }
```

Функция возвращает логическое значение, то есть производит проверку свойств числа **a.** Внешний цикл выделяет в нем последовательность цифр, очередная цифра хранится в переменной **k.** Внутренний цикл проверяет свойства делимости этой цифры. Причем как после внешнего, так и после внутреннего цикла проверяется условие «естественного» выхода из цикла: похоже, проверяются свойства всеобщности или существования. Внутренний цикл, проверяющий на цифру **k** на делимость в диапазоне от 2 до **k-1,** определяет, является ли **k** простым. Тогда оба цикла проверяют один из 3 возможных вариантов: все цифры числа – простые, все цифры числа – не простые, в числе есть те и другие цифры.

Для точной подгонки результата нужно проследить цепочку операторов **break** и условий, по которым они происходят. Самый внутренний **break** происходит при обнаружении делителя цифры (**k** – не простое). Следующий **break** происходит, если условие «естественного» выхода не было достигнуто, то есть был предыдущий **break** и выход из внешнего цикла происходит по тому же условию (**k** – не простое). Но последнее условие (**n==0**) является условием «естественного» завершения этого же цикла, то есть проверяется отсутствие предыдущего **break** по не простой цифре. Таким образом, функция проверяет, **все ли цифры числа являются простыми**. Поскольку цифры 0,1 внутренним циклом не проверяются, для них сделано исключение.

```
//-----------------------------------------------------21-08.cpp
//-----------------------------------------------------1
 int F1(int n){
 for ( int i=2; n % i !=0; i++);
 return i; }
 //-----------------------------------------------------2
 int F2(int n1, int n2){
 for ( int i=n1-1; !(n1 % i ==0 && n2 % i ==0); i--);
 return i; }
 //-----------------------------------------------------3
 int F3(int n1, int n2){
 int i = n1; if (i < n2) i=n2;
 for (; !(i % n1 ==0 && i % n2 ==0); i++);
 return i; }
 //-----------------------------------------------------4
 int F4(int a){
 for ( int n=2; n<a; n++)
     { if (a%n==0) break; }
 if (n==a) return 1;
 return 0;}
 //-----------------------------------------------------5
 int F5(int a){
 for ( int s=0,n=2; n<a; n++)
     { if (a%n==0) s++; }
 if (s==0) return 1;
 return 0;}
```

```
//-----------------------------------------------------6
int F6(int a, int b){
for ( int n=a; n%a!=0 || n%b!=0; n++);
return n; }
//-----------------------------------------------------7
int F7(int a, int b){
for ( int n=a-1; a%n!=0 || b%n!=0; n--);
return n; }
//-----------------------------------------------------8
int F8(int a){
int n,k,s;
for (n=a, s=0; n!=0; n=n/10)
    { k=n%10; s=s+k;}
return s;}
//-----------------------------------------------------9
int F9(int a){
int n,k,s;
for (n=a, s=0; n!=0; n=n/10)
    { k=n%10; if (k>s) s=k;}
return s;}
//-----------------------------------------------------10
int F10(int a){
int n,k,s;
for (n=a, s=0; n!=0; n=n/10)
    { k=n%10; s=s*10+k;}
return s;}
//-----------------------------------------------------11
void F11(){
int a,n,k,s;
    for (a=10; a<30000; a++){
    for (n=a, s=0; n!=0; n=n/10)
        { k=n%10; s=s+k;}
        if (a==s*s*s) printf("%d\n",a);
    } }
//-----------------------------------------------------12
void F12(int a, int A[10]){
int i,n;
for (i=0, n=a; n!=0; i++, n=n/10);
for (A[i--]=-1, n=a; n!=0; i--, n=n/10)
    A[i]=n % 10;
}
//-----------------------------------------------------13
void F13(int v, int A[], int m){
int i,n,a;
    for (i=0,a=2; a<v && i<m-1 ; a++){
    for (n=2; n<a; n++)
        { if (a%n==0) break; }
    if (n==a) A[i++]=a;
    }
A[i]=0; }
//-----------------------------------------------------14
void F14(int v, int A[], int m){
int i,n,a,j;
    for (i=0,a=2; a<v && i<m-1 ; a++){
    for (j=0; j<i; j++)
        { if (a%A[j]==0) break; }
    if (j==i) A[i++]=a;
    }
A[i]=0; }
//-----------------------------------------------------15
void F15(int val, int A[], int n){
int m,i;
    for (i=0; i<n-1 && val !=1; i++){
```

```
    for (m=2; val % m !=0; m++);
    val /= m; A[i] = m;
    }
A[i] = 0;}
//-----------------------------------------------------16
int F16(int c[], int n){
int i,j;
for (i=0; i<n-1; i++)
    for (j=i+1; j<n; j++)
        if (c[i]==c[j]) return i;
return -1; }
//-----------------------------------------------------17
int F17(int n)
{ int k,m;
for (k=0, m=1; m <= n; k++, m = m * 2);
return k-1; }
//-----------------------------------------------------18
void F18(int c[], int n)
{ int i,j,k;
for (i=0,j=n-1; i < j; i++,j--)
    { k = c[i]; c[i] = c[j]; c[j] = k; }
}
//-----------------------------------------------------19
int F19(int c[], int n)
{ int i,j,k1,k2;
    for (i=0; i<n; i++){
    for (j=k1=k2=0; j<n; j++)
        if (c[i] != c[j])
        { if (c[i] < c[j]) k1++; else k2++; }
    if (k1 == k2) return i;
    }
return -1; }
//-----------------------------------------------------20
int F20(int c[], int n)
{ int i,j,m,s;
    for (s=0, i=0; i < n-1; i++){
    for (j=i+1, m=0; j<n; j++)
```

2.2. Итерационные циклы

Для заданного варианта написать функцию вычисления суммы ряда. для диапазона значений 0.1 .. 0.9 и шага 0.1 изменения аргумента вычислить значения суммы ряда и контрольной функции, к которой он сходится, с точностью до 4 знаков после запятой.

Для вариантов 6-8 использовать значение sin и cos на предыдущем шаге для вычисления значений на текущем шаге по формулам:

$$\sin(nx) = \sin((n-1)x)\cos(x) + \cos((n-1)x)\sin(x)$$
$$\cos(nx) = \cos((n-1)x)\cos(x) - \sin((n-1)x)\sin(x)$$

Вариант	Ряд	Функция
1	$1 - x^2/2! + \ldots + (-1)^n x^{2n}/(2n)!$	cos(x)
2	$z=((x-1)/(x+1))\quad (2/1)z^3 + (2/3)z^5 + \ldots + (2/2n-1)z^{2n-1}$	ln(x)

3	$x + \frac{1}{3}x + \frac{3}{15}x + \ldots + 2^{2n}(2^{2n}-1)B_n x^{2n-1}/(2n)!$ (ряд с n=1)	tg(x)
4	$x + \frac{1}{2}\frac{3}{3}x + \ldots + \frac{1\,3\,5\ldots(2n-1)}{2\,4\,6\ldots2n(2n+1)}x^{2n+1}$	arcsin(x)
5	$1 + x\ln 3 + (x\ln 3)^2/2! + \ldots + (x\ln 3)^n/n!$	3^x
6	$1 + \frac{1}{2}x + \frac{1}{2}\frac{1}{4}x^2 - \ldots + (-1)^n\frac{1\,1\,3\ldots(2n-3)}{2\,4\,6\ldots2n}x^n$	$(1+x)^{0.5}$
7	$\sin(x) - \sin(2x)/2 + \ldots + (-1)^n\sin(nx)/n$	x / 2
8	$-\cos(x) + \cos(2x)/2^2 + \ldots + (-1)^n\cos(nx)/n^2$	$0.25(x - PI^2/3)$
9	$\frac{1}{x} - \frac{1}{x}\left(x + \frac{1}{3}x^3 + \frac{2}{45}x^5 + \ldots + 2^{2n}B_n x^{2n-1}/(2n)!\right)$	ctg(x)
10	$(x-1)/x + (x-1)^2/2x + (x-1)^3/3x + \ldots + (x-1)^n/nx$	ln(x)
11	$-x/2 - x^2/12 - x^4/45 - \ldots - 2^{2n}(2^{2n}-1)B_n x^{2n}/n(2n)!$	ln(cos(x))
12	$x + x^3/3! + x^5/5! + \ldots + x^{2n+1}/(2n+1)!$	sh(x)
13	$1 + x^2/2! + x^4/2! + \ldots + x^{2n}/2n!$	ch(x)
14	$x - x^3/3 + x^5/5 + \ldots + (-1)^n x^{2n-1}/(2n-1)$	arctg(x)
15	$PI/2 - 1/x + 1/3x^3 - 1/5x^5 + \ldots + (-1)^{n+1}/(2n+1)x^{2n+1}$	arctg(x)
16	$1 - \frac{1}{2}x^2 + \frac{1\,3}{2\,4}x^4 - \frac{1\,3\,5}{2\,4\,6}x^6 + \frac{1\,3\,5\,7}{2\,4\,6\,8}x^8$	$1/(1+x)^{0.5}$
17	$1 - 2x^2 + 3x^4 - 4x^6 + 5x^8 + \ldots + (-1)^n(n+1)x^{2n}$	$1/(1+x)^2$
18	$\cos(x) + \cos(3x)/3 + \ldots + \cos((2n-1)x)/(2n-1)$	0.5ln(ctg(x/2))

```
19 cos(x) - cos(2x)/2 +...+ (-1) cos(nx)/n                ln(2cos(x/2))
_____________________________________________________________________
                        3                          3
20 sin(x) + sin(3x)/3 +...+ sin((2n-1)x)/(2n-1)          (PI/8)x(PI-x)
_____________________________________________________________________
```

Тестовые задания

Восстановить в общем виде формулу степенного ряда, вычисляемого в данной функции.

Пример выполнения тестового задания.

```
for (s=0, sn = 1 n=2; fabs(sn) > eps; n+=2)
      { s += sn;
      sn= sn * x * x / (n*(n+1)); }
```

Для получения аналитической зависимости необходимо восстановить последовательность значений степеней **x** и произведений целых коэффициентов, составляющих факториалы, либо сокращающихся при переходе от шага к следующему. После получения явно наблюдаемой зависимости необходимо перевести ее в аналитический вид **от натурального n.**

N	sn до	Коэффициент	sn после
2	1	x^2 / (2*3)	x^3*(2*3)
4	x^2(2*3)	x^2 / (4*5)	x^5/(2*3*4*5)
6	x^4(2*3*4*5)	x^2 / (6*7)	x^7/(2*3*4*5*6*7)
M	x^2m/(2*m-1)!		

В данном примере накапливаются четные степени **x** и нечетные факториалы (обратите внимание, что **n** в самой программе меняется через 2, а ряд выражен через натуральное **m**).

```
//-----------------------------------------------------22-06.cpp
//-------------------------------------------------1
 double s,sn; int n;
 for (s=0, sn = 1, n=1; fabs(sn) > eps; n++)
 { s += sn;
 sn= - sn * x / n; }
 //-------------------------------------------------2
 for (s=0, sn = x, n=1; fabs(sn) > eps; n++)
 { s += sn;
 sn= - sn * x / n; }
 //-------------------------------------------------3
 for (s=0, sn = x; n=1; fabs(sn) > eps; n+=2)
 { s += sn;
 sn= sn * x * x / (n*(n+1) ); }
 //-------------------------------------------------4
 for (s=0, sn = x, n=1; fabs(sn) > eps; n+=2)
 { s += sn;
 sn= sn * x / (n *(n+1) ); }
 //-------------------------------------------------5
 for (s=0, sn = x, n=1; fabs(sn) > eps; n++)
 { s += sn;
 sn= sn * x * (2*n) / (2*n-1); }
 //-------------------------------------------------6
 for (s=0, sn = x, n=1; fabs(sn) > eps; n+=2)
```

```
{ s += sn;
sn= sn * x *x * n / (n+1); }
//---------------------------------------------7
for (s=0, sn = x, n=1; fabs(sn) > eps; n++)
{ s += sn;
sn= sn * x * x * (2*n-1) / (2*n+1); }
//---------------------------------------------8
for (s=0, sn = x, n=2; fabs(sn) > eps; n+=2)
{ s += sn;
sn= sn * x *x * (n -1) / (n+1); }
//---------------------------------------------9
for (s=0, sn = 1, n=1; fabs(sn) > eps; n++)
{ s += sn;
int nn = 2*n-2; if (nn==0) nn=1;
sn= sn * x * x * nn / (2*n); }
//---------------------------------------------10
for (s=0, sn = 1, n=1; fabs(sn) > eps; n+=2)
{ s += sn;
int nn = n-1; if (nn==0) nn=1;
sn= sn * x *x * nn / (n+1);}
```

2.4. Работа со строками

1. Выполнить сортировку символов в строке. Порядок возрастания "весов" символов задать таблицей вида `char ORD[] = "АаБбВвГгДдЕе1234567890";` Символы, не попавшие в таблицу, размещаются в конце отсортированной строки.

2. В строке, содержащей последовательность слов, найти конец предложения, обозначенный символом "точка". В следующем слове первую строчную букву заменить на прописную.

3. В строке найти все числа в десятичной системе счисления, сформировать новую строку, в которой заменить их соответствующим представлением в шестнадцатеричной системе.

4. Заменить в строке принятое в Си обозначение символа с заданным кодом (например, \101) на сам символ (в данном случае - A).

5. Переписать в выходную строку слова из входной строки в порядке возрастания их длины.

6. Преобразовать строку, содержащую выражение на Си с операциями (=,==,!=,a+=,a-=), в строку, содержащую эти же операции с синтаксисом языка Паскаль (:=,=,#,a=a+,a=a-).

7. Удалить из строки комментарии вида "/* ... */". Игнорировать вложенные комментарии.

8. Заменить в строке символьные константы вида 'A' на соответствующие шестнадцатеричные (т.е. 'A' на 0x41).

9. Заменить в строке последовательности одинаковых символов (не пробелов) на десятичное число, состоящее из двух десятичных цифр и соответствующее их количеству (т.е. « abcdaaaaa xyznnnnnnn » на «abcd5a xyz7n »).

10. Найти в строке два одинаковых фрагмента (не включающих в себя пробелы) длиной более 5 символов и возвратить индекс начала первого из них (т.е. для «aaaaaabcdefgxxxxxxbcdefgwwwww» вернуть n=6 - индекс начала «bcdefg»).

11. Оставить в строке фрагменты, симметричные центрального символа, длиной более 5 символов (например, «dcbabcd»), остальные символы заменить на пробелы.

12. Найти во входной строке самую внутреннюю пару скобок {...} и переписать в выходную строку содержащиеся между ними символы. Во входной строке фрагмент удаляется.

13. Заменить в строке все целые числа соответствующим повторением следующего за ними символа (например "abc5xacb15y" - " abcxxxxxacbyyyyyyyyyyyyyyy ").

14. "Перевернуть" в строке все слова. (Например: "Жили были дед и баба" - "илиЖ илиб дед и абаб").

15. Функция переписывает строку. Если она находит в строке число, то вместо него она переписывает в выходную строку соответствующее по счету слово из входной строки. (например, "aaa bb1bb cc2cc" - "aaa bbaaabb ccbb1bbcc").

Тестовые задания

Содержательно определите действие, производимое над строкой. Напишите вызов функции (входные неизменяемые строки могут быть представлены фактическими параметрами – строковыми константами).

Пример оформления тестового задания.

```
//-----------------------------------------------------24-12.cpp
int F(char c[]){
 for (int i=0,ns=0; c[i]!=0; i++)
        if (c[i]!=' ' && (c[i+1]==' ' || c[i+1]==0)) ns++;
 return ns;}
 #include <stdio.h>
 void main(){ printf("words=%d\n",F("aaaa bbb ccc dddd"));}   // Выведет - 4
```

Функция работает со строкой, поскольку в качестве параметра получает массив символов, которую просматривает до обнаружения символа конца строки. Переменная **ns** является счетчиком. Условие, выполнение которого увеличивает счетчик – текущий символа не является пробелом, а следующий – пробел, либо конец строки. Это условие обнаруживает конец слова. Таким образом, программа подсчитывает количество слов в строке, разделенных пробелами.

```
//-----------------------------------------------------24-13.cpp
//------------------------------------------------ 1
 void F1(char c[])
 { int i,j;
 for (i=0; c[i] !='\0'; i++);
 for (j=0,i--; i>j; i--,j++)
        { char s; s=c[i]; c[i]=c[j]; c[j]=s; }}
 //------------------------------------------------ 2
 int F2(char s)
 { if (s >='0' && s <='9') return s - '0';
 else return -1; }
 //------------------------------------------------ 3
 void F3(char c[]){
 for ( int i=0; c[i] !='\0'; i++)
        if (c[i] >='a' && c[i] <='z')
                c[i] += 'A' - 'a'; }
 //------------------------------------------------ 4
 int F4(char c[])
```

```
{ int i,old,nw;
  for (i=0, old=0, nw=0; c[i] !='\0'; i++) {
        if (c[i]==' ') old = 0;
                else { if (old==0) nw++; old=1; }
        if (c[i]== '\0') break;
        }
return nw; }
//------------------------------------------------ 5
void F5(char c[]){
for ( int i=0, j=0; c[i] !='\0'; i++)
        if (c[i] !=' ') c[j++] = c[i];
c[j] = '\0'; }
//------------------------------------------------ 6
void F6(char c[], int nn)
{ int k,mm;
for (mm=nn, k=0; mm !=0; mm /=10 ,k++);
for (c[k--]='\0'; k>=0; k--)
        { c[k]= nn % 10 + '0'; nn /=10; }
}
//------------------------------------------------ 7
int F7(char c[])
{ int i,s;
for (i=0; c[i] !='\0'; i++)
        if (c[i] >='0' && c[i]<='7') break;
for (s=0; c[i] >='0' && c[i] <='7'; i++)
        s = s * 8 + c[i] - '0';
return s; }
//------------------------------------------------ 8
int F8(char c[])
{ int n,k,ns;
 for (n=0,ns=0; c[n] !='\0'; n++) {
        for (k=0; n-k !=0 && c[n+k] !='\0'; k++)
                if (c[n-k] != c[n+k]) break;
        if (k >=3) ns++;
        }
return ns; }
//------------------------------------------------ 9
int F9(char c1[],char c2[])
{ int i,j;
  for (i=0; c1[i] !='\0'; i++) {
        for (j=0; c2[j] !='\0'; j++)
                if (c1[i+j] != c2[j]) break;
        if (c2[j] =='\0') return i;
        }
return -1;}
//------------------------------------------------ 10
char F10(char c[])
{ char m,z; int n,s,i;
  for (s=0,m='A'; m <='Z'; m++) {
        for (n=0, i=0; c[i] !='\0'; i++)
                if (c[i]==m) n++;
                if (n > s) { z=m; s=n; }
        }
return z; }
//------------------------------------------------ 11
void F11(char c[], double x)
{ int i;  x-=(int)x;
 for (c[0]='.', x -= (int)x, i=1; i<6; i++) {
        x *= 10.; c[i]=(int)x + '0'; x -= (int)x; }
c[i]='\0'; }
//------------------------------------------------ 12
int F12(char c[]){
for (int i=0; c[i]!=0; i++){
```

```
        if (c[i]==' ') continue;
        for (int j=i+1; c[j]==c[i]; j++);
        for (; c[j]!=0; j++){
                for (int k=0; i+k<j && c[i+k]==c[j+k]; k++);
                if (k>=4) return i;
        }} return -1; }
//--------------------------------------------- 13
void F13(char c[])
{ int i,j,cm;
 for (i=j=cm=0; c[i] !='\0'; i++) {
          if (c[i]=='*' && c[ i+1]=='/') { cm--, i++; continue; }
          if (c[i]=='/' && c[ i+1]=='*') { cm++, i++; continue; }
          if (cm==0) c[j++ ] = c[i];
          }
c[j]=0; }
```

2.5. Сортировка и поиск

Алгоритм сортировки реализовать в виде функции, возвращающей в качестве результата характеристику трудоемкости алгоритма (например, количество сравнений). Если имеется базовый алгоритм сортировки (для Шелла - «пузырек», для Шейкер - «пузырек», для вставки с двоичным поиском - погружение), то аналогично оформить базовый алгоритм и произвести сравнение эффективности.

1. Сортировка вставками, место помещения очередного элемента в отсортированную часть производить с помощью двоичного поиска. Двоичный поиск оформить в виде отдельной функции.

2. Сортировка Шелла. Частичную сортировку с заданным шагом, начиная с заданного элемента оформить в виде функции. Алгоритм частичной сортировки - вставка погружением.

3. Сортировка разделением. Способ разделения: вычислить среднее арифметическое всех элементов массива и относительно этого значения разбить массив на две части (с использованием вспомогательных массивов).

4. Шейкер-сортировка. Процесс движения в прямом и обратном направлении реализовать в виде одного цикла, используя параметр - направление движения (+1/-1) и меняя местами нижнюю и верхнюю границы просмотра.

5. «Быстрая» сортировка с итерационным циклом вычисления медианы. Для заданного интервала массива, в котором производится разделение, ищется медиана обычным способом. Затем выбирается та часть интервала между границей и медианой, в которой находится середина исходного интервала, и процесс повторяется.

6. Сортировка циклическим слиянием с использованием одного выходного и двух входных массивов. Для упрощения алгоритма и разграничения сливаемых групп в последовательности в качестве разделителя добавить «очень большое значение» (MAXINT).

7. Сортировка разделением. Способ разделения: интервал между минимальным и максимальным значениями элементов массива разбить пополам и относительно этого значения разбить массив на две части (с использованием вспомогательных массивов).

8. Простое однократное слияние. Разделить массив на n частей и отсортировать их произвольным методом. Отсортированный массив получить путем однократного слияния упорядоченных частей. Для извлечения очередных

элементов из упорядоченных массивов использовать массив из n индексов (по одному на каждый массив).

9. Сортировка подсчетом. Выходной массив заполняется значениями «-1». Затем для каждого элемента определяется его место в выходном массиве путем подсчета количества элементов строго меньших данного. Естественно, что все одинаковые элементы попадают на одну позицию, за которой следует ряд значений «-1». После чего оставшиеся в выходном массиве позиции со значением «-1» заполняются копией предыдущего значения.

10. Сортировка выбором. Выбирается минимальный элемент в массиве и запоминается. Затем удаляется, а все последующие за ним элементы сдвигаются на один влево. Сам элемент заносится на освободившуюся последнюю позицию.

11. Сортировка вставками. Берется очередной элемент и извлекается из массива. Затем от начала массива ищется первый элемент, больший данного. Все элементы, от найденного до очередного сдвигаются на один вправо и на освободившееся место помещается очередной элемент. (Поиск места включения от начала упорядоченной части).

12. Сортировка выбором. Выбирается минимальный элемент в массиве, переносится в выходной массив на очередную позицию и заменяется во входном на «очень большое значение» (MAXINT).

13. Сортировка Шелла. Частичную сортировку с заданным шагом, начиная с заданного элемента оформить в виде функции. Алгоритм частичной сортировки - обменная (методом «пузырька»).

14. Сортировка выбором. Выбирается минимальный элемент в массиве, переносится в выходной массив на очередную позицию. Во входном массиве все элементы от следующего за текущим до конца сдвигаются на один влево.

15. Сортировка «хитрая». Из массива путем однократного просмотра выбирается последовательность элементов, находящихся в порядке возрастания, переносятся в выходной массив и заменяются во входном на «-1». Затем оставшиеся элементы включаются в полученную упорядоченную последовательность методом погружения.

16. Оптимизированный двоичный поиск. В процессе деления выбирается не середина интервала, а значение, вычисленное из предположения о линейном возрастании значений элементов массива в текущем интервале поиска. Сравнить эффективности разработанного и базового алгоритмов на массивах с резко неравномерным возрастанием значений (например, 1,2,2,3,4,25).

Тестовые задания

По тексту программы определите алгоритм сортировки. Определите «смысл» отдельных переменных и назначение циклов.

```
//-----------------------------------------------------25-16.cpp
//------------------------------------------------ 1
 void F1(int in[],int n)
 { int i,j,k,c;
 for (i=1; i<n; i++){
     for (k=i; k !=0; k--)
         { if (in[k] > in[k-1]) break;
         c=in[k]; in[k]=in[k-1]; in[k-1]=c;
```

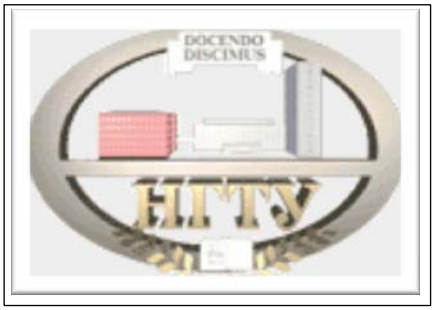
```
        }
    }}
//------------------------------------------------ 2
void F2(int in[],int out[],int n)
{ int i,j ,cnt;
for (i=0; i< n; i++) {
    for ( cnt=0,j=0; j<n; j++)
        if (in[j] > in[i]) cnt++;
    else
        if (in[j]==in[i] && j>i) cnt++;
    out[cnt]=in[i];
    } }
//------------------------------------------------ 3
void F3(int in[],int n)
{ int a,b,dd,i,lasta,lastb,swap;
for (a=lasta=0, b=lastb=n, dd=1; a < b; dd = !dd, a=lasta, b=lastb){
      if (dd){
            for (i=a,lastb=a; i<b; i++)
                  if (in[i] > in[i+1]){
                        lastb = i;
                        swap = in[i]; in[i]=in[i+1]; in[i+1]=swap;
                  }
        }
        else {
            for (i=b,lasta=b; i>a; i--)
                  if (in[i-1] > in[i]){
                  lasta = i;
                  swap = in[i]; in[i]=in[i-1]; in[i-1]=swap;
            }}}}
//------------------------------------------------ 4
int find(int out[],int n, int val);
// Двоичный или линейный поиск расположения значения val
// в массиве out[n]
void F4(int in[], int n){
int i,j,k;
for (i=1; i<n; i++)
    { int c; c = in[i]; k = find(in,i,c);
    for (j=i; j!=k; j--) in[j] = in[j-1];
    in[k] = c; }
}
//------------------------------------------------ 5
void F5(int in[], int n){
int i,j,c,k;
for (i=0; i < n-1; i++){
    for (j=i+1,c=in[i],k=i; j<n; j++)
            if (in[j] > c) { c = in[j]; k=j; }
    in[k] = in[i]; in[i] = c;
    }}
//------------------------------------------------ ---6
void F6(int A[], int n){
int i,found;
do { found =0;
```

```
        for (i=0; i<n-1; i++)
        if (A[i] > A[i+1])
                { int cc; cc = A[i]; A[i]=A[i+1]; A[i+1]=cc;
                found++;
                }
        } while(found !=0); }
//------------------------------------ ----------7
void sort(int a[], int n); //любая сортировка одномерного массива
#define MAXINT 1000
int A[100], B[10][10];
void F7(){
int i,j;
for (i=0; i<100; i++) B[i/10][i%10]=A[i];
for (i=0; i<10; i++) sort(B[i],10);
for (i=0; i<100; i++){
        int k;
        for (k=0, j=0; j<10; j++)
                if (B[j][0] < B[k][0]) k=j;
        A[i] = B[k][0];
        for (j=1; j<10; j++) B[k][j-1]=B[ k] [j];
        B[k][9]=MAXINT;
        }}
//-------------------------------------------------8
void F8(int in[], int a, int b){
int i,j,mode;
if (a >=b) return;
for (i=a, j=b, mode=1; i < j; mode >0 ? j-- : i++)
      if (in[i] > in[j]){
            int c = in[i]; in[i] = in[j]; in[j]=c;
            mode = -mode;
      }
F8(in,a,i-1); F8(in,i+1,b); }
//-------------------------------------------------9
void F9(int A[], int n){
int i,b,b1;
for (b=n-1; b!=0; b=b1) {
        b1=0;
        for (i=0; i<b; i++)
            if (A[i] > A[i+1]) {
                  int cc = A[i]; A[i]=A[i+1]; A[i+1]=cc;
                  b1=i;
            }}}
//-------------------------------------------------10
void F10(int A[], int B1[], int B2[], int n){
int i,i1,i2,s,a1,a2,a,k;
for (s=1; s!=n; s*=2){
      for (i=0; i<n/2; i++)
            { B1[i]=A[i]; B2[i]=A[i+n/2]; }
      i1=i2=0; a1=a2=MAXINT;
      for (i=0,k=0; i<n; i++)
            {
            if (a1==MAXINT && a2==MAXINT && i1==s && i2==s)
```

```
            k+=s,i1=0,i2=0;
if (a1==MAXINT && i1!=s) a1=B1[k+i1],i1++;
if (a2==MAXINT && i2!=s) a2=B2[k+i2],i2++;
if (a1<a2)a=a1,a1=MAXINT;
else a=a2,a2=MAXINT;
A[i]=a;
}}}
```

Новосибирский государственный технический университет

Кафедра вычислительной техники



Пояснительная записка к Расчётно-графической работе.

Вариант 2

«Использование графической библиотеки “Graphics.h”»

Группа: АВТ-009
Студент: Мельников Т. А.

Новосибирск
2010.

1. Задание

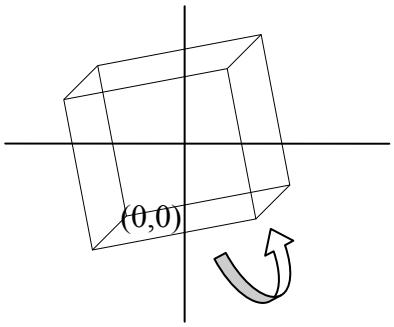
Вариант 2. По экрану движется вращающийся куб, изображаемый в виде граней. Вращение управляется с клавиатуры.

2. Основные идеи и методы решения

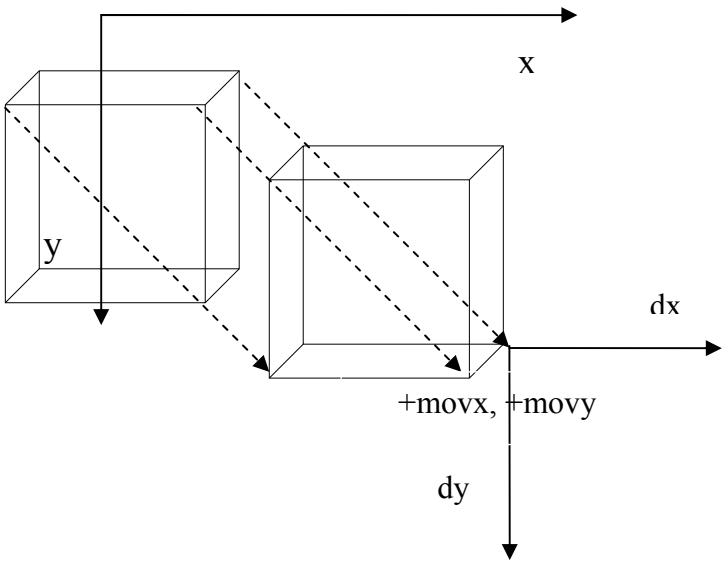
Для реализации движения куба необходимо решить следующие проблемы:

- движение точек в системе координат относительно экрана;
- вращение куба относительно точки и параллельный перенос вершин куба из начала координат;
- изменение направления движения куба и отскока его от границ экрана.

В программе используется две прямоугольные системы координат. При реализации вращательного используется система с началом координат в центре куба.



Затем моделируется движение центра куба в основной системе координат, с соответствующим переносом вершин.



В соответствии с выбранной геометрической моделью в программе реализуюися различные виды движения:

1) ***Вращательное движение*** достигается за счет перемножения каждого трехмерного вектора координаты на матрицу поворота. Для того, чтобы куб вращался вокруг своего центра, вектор координату перемножаем по очереди на матрицу поворота вокруг каждой оси (OX, OY, OZ). Вращение происходит вокруг точки начала координат.
2) ***Линейное движение.*** Т.к. можно вращать куб только вокруг начала координат, то трехмерные координаты не меняются (отсутствует линейное движение) и, следовательно, проецируемые двумерные координаты всегда стоят на месте. Линейное движение происходит за счет прибавления к двумерной координате постоянно увеличивающейся (уменьшающейся) на dx, dy переменной.
3) ***Контроль движения.*** Слежение за процессом движения производится рядом условий, в которых задействованы координаты вершин куба. Условия необходимы для удержания фигуры в ограниченной на экране области. Кроме того, используются две переменные, содержащие информацию о направлении движения и одна для управления направления вращения.

3. Функциональное описание программы

Программа содержит следующие данные.

```
#define A 50      // половина грани куба
double X3[8],Y3[8],Z3[8], X2[8],Y2[8];
                  // глобальные массивы трехмерной и двумерной координаты
double L=0.5, fi=45*(3.14159265358/180.0); //fi угол проецирования координаты
int dx=2,dy=2;          // скорость линейного движения
int k=0;                // счетчик градусов
char c;                 // направление вращательного движения
long double degree;     // градус поворота
```

Программа состоит из главного цикла void main() и из функций cube(), line(),rotate (long double degree), projection (double L, double fi)

- *cube()* – задование трехмерных координат куба;
- *line()* – прорисовка куба;
- *rotate (long double degree)* – вращение куба;
- *projection (double L, double fi)* – проецирование трехмерных координат.

Главный цикл. В теле главного цикла происходит определение направления вращения куба, вызов функций *rotate (long double degree), projection (double L, double fi), cube(), line(),* проверка условий отскакивания от стенок и изменение направления движения при совпадение условия.

```
void main (){
```

```
      int gdriver=DETECT, gmode=VGAHI, errorcode;
      initgraph(&gdriver, &gmode, "e:\\bc31\\bgi");
      errorcode = graphresult();
      if (errorcode != grOk)
      {
            printf("Graphics error: %s\n", grapherrormsg(errorcode));
            printf("Press any key to halt...");
            getch();
            return;
      }

      double L=0.5, fi=45*(3.14159265358/180.0);
      //fi угол проецирования, L коэффициент проецирования
      int maxx=getmaxx();
      int maxy=getmaxy();
      int i;
      int dx=2,dy=2;          // скорость линейного движения
      int movx=100, movy=100 ;
      long double degree;     // градус поворота

      cube();
      char c;              // направление вращательного движения
      while(c!='r') {      // пока не нажата клавиша "r"

      c=getch();
      if (c=='a') {degree= 3.14159265358/180.0;}
      //нажата "a" по часовой стрелке
      if (c=='d') {degree= -3.14159265358/180.0;}
      // нажата "d" против часовой стрелки
      rotate (degree);

      projection (L,fi);       //проецирование трехмерных координат
      for (i=0;i<8;i++) {
            X2[i]=X2[i]+movx;     // "вынос" куба из начала координат
            Y2[i]=Y2[i]+movy;
            if ((X2[i]>=maxx || X2[i]<=0) && (X2[i]+3*dx>=maxx ||
X2[i]+3*dx<=0)) // условие отскока от горизонтальной стенки
                  { dx=-dx;}
            if ((Y2[i]>=maxy || Y2[i]<=0) && (Y2[i]+3*dy>=maxy ||
Y2[i]+3*dy<=0)) // условие отскока от вертикальной стенки
                  { dy=-dy;}
      }
      movx+=dx;   //процесс смещения координат куба
      movy+=dy;
      setfillstyle(1,BLACK);
      bar(0,0,maxx,maxy); // затирание старого куба
      setcolor (WHITE);
      line ();                //создание нового куба
      delay(7);
}
}
```

Функция задания координат *cube()* в качестве параметров получает глобальные массивы трехмерных координат X3, Y3, Z3 и половину длины грани A. Координаты расположены вокруг начала координат.

```
void cube () {
      X3[0]=-A; Y3[0]=-A; Z3[0]=-A;      X3[4]=-A; Y3[4]=-A; Z3[4]=A;
      X3[1]=A; Y3[1]=-A; Z3[1]=-A;    X3[5]=A; Y3[5]=-A; Z3[5]=A;
      X3[2]=A; Y3[2]=A; Z3[2]=-A;  X3[6]=A; Y3[6]=A; Z3[6]=A;
      X3[3]=-A; Y3[3]=A; Z3[3]=-A;    X3[7]=-A; Y3[7]=A; Z3[7]=A;
}
```

Функция прорисовки куба *line()* в качестве параметров получает глобальные массивы двумерных координат X2, Y2.

```
void line () {
      for (int i=0;i<7;i++) { if (i==3) continue;
      line (X2[i],Y2[i],X2[i+1],Y2[i+1]);}
      for(i=0;i<4;i++){
            line (X2[i],Y2[i],X2[i+4],Y2[i+4]);}
      line (X2[0],Y2[0],X2[3],Y2[3]);
      line (X2[4],Y2[4],X2[7],Y2[7]);}
```

Функция вращения куба вокруг точки начала координат *rotate (long double degree)* в качестве параметров получает значения градуса поворота и глобальные массивы трехмерных координат. Производит перемножение вектора координаты на матрицу поворота.

```
void rotate (long double degree) {
      for(int i=0;i<8;i++) {                          //вращение вокруг оси OX
            Y3[i]= Y3[i]*cos(degree)-Z3[i]*sin(degree);
            Z3[i]= Y3[i]*sin(degree)+Z3[i]*cos(degree);
      }
      for(i=0;i<8;i++) {                              //вращение вокруг оси OZ
            X3[i]= X3[i]*cos(degree)-Y3[i]*sin(degree);
            Y3[i]= X3[i]*sin(degree)+Y3[i]*cos(degree);
      }
      for(i=0;i<8;i++) {                              // вращение вокруг OY
            X3[i]= X3[i]*cos(degree)-Z3[i]*sin(degree);
            Z3[i]= X3[i]*sin(degree)+Z3[i]*cos(degree);
      }
}
```

Функция проецирования трехмерных координат *projection (double L, double fi)* в качестве параметров передается угол и коэффициент проецирования. Использует глобальные массивы трехмерных и двумерных координат.

```
void projection (double L, double fi){
      for (int i=0;i<8;i++) {
            X2[i]=X3[i]+Z3[i]*L*cos(fi);
            Y2[i]=Y3[i]+Z3[i]*L*sin(fi);
      }
}
```

Информатика-1. Практические занятия.

Контрольная работа №1. Арифметические задачи

Любая целочисленная денежная сумма n>7 может быть выдана без сдачи “трешками” и “пятерками”. Программа находит эти числа для заданного n.

Найти сумму цифр заданного числа.

Найти максимальную цифру в заданном числе.

Задумано целое число. Известны k,m,n – остатки от деления числа на 3,5,7. Найти это число.

Найти наименьшее общее кратное для двух заданных чисел. Проверить, являются ли эти числа взаимно простыми.

Найти наибольший общий делитель для двух заданных чисел. Проверить, являются ли эти числа взаимно простыми.

На заданном интервале найти все числа, цифры которых находятся в строго возрастающем порядке (например, 532).

На заданном интервале найти все числа, сумма цифр которого в кубе равна самому этому числу (например, 512=(5+1+2)^3=8^3=512).

Число Армстронга – такое число из k цифр, для которого сумма k-х степеней его цифр равна самому этому числу, например $153=1^3+5^3+3^3$. Найти все трехзначные числа Армстронга.

Найти все двухзначные (трехзначные) числа, которые совпадают с последними цифрами своих квадратов, например, $25^2=625$, $76^2=5676$.

Утверждается, что разность любого натурального числа и суммы его цифр кратна 9. Проверить это факт для всех чисел заданного диапазона.

Число делится на 11, если разность между суммой цифр на четных и нечетных местах делится на 11. Проверить это факт для всех чисел заданного диапазона.

Число называется совершенным, если оно равно сумме всех своих делителей, например, 6=1+2+3, 28=1+2+4+7+14. Найти все совершенные числа в заданном интервале.

Найти все числа в заданном диапазоне, которые делятся на любую из своих цифр.

Контрольная работа №2. Задачи на массивах

1. Найти в массиве минимальный и максимальный элементы и поменять их местами.
2. Найти в массиве минимальный элемент и удалить.
3. Найти в массиве все пары одинаковых, рядом стоящих элементов и удалить.
4. Удалить из массива все отрицательные элементы.
5. Найти среднее арифметическое элементов массива и элемент, наиболее близкий к нему.
6. Найти среднее между минимальным и максимальным элементом массива и элемент, наиболее близкий к нему.
7. Найти “центр тяжести” массива – разбить массив на две части, суммы значений элементов справа и слева наиболее близки (минимум модуля разности сумм).
8. В упорядоченный массив включить новый элемент с сохранением порядка.
9. Найти в массиве все пары одинаковых элементов (не обязательно рядом стоящих) и вывести.
10. Найти в массиве все пары взаимно простых элементов (не обязательно рядом стоящих) и вывести.
11. Найти в массиве все последовательности возрастающих элементов и копировать в выходной массив.
12. Удалить из массива все не простые числа (оставить простые).
13. Найти в массиве все пары элементов (не обязательно рядом стоящих), у которых последняя цифра первого элемента равна первой цифре второго и вывести.

14. В массиве, содержащем нечетное количество элементов, найти медиану – значение, для которого количество больших элементов равно количеству меньших. Учесть тот факт, что при наличии равных элементов медиана не может быть определена.
15. Определить количество неповторяющихся значений элементов массива (например 1,7,2,8,1,1,1,2,5,7,7 -> 2).
16. Найти в массиве минимальный и максимальный элементы. Вывести последовательность значений из этого диапазона, не встречающихся в данном массиве.
17. Выделить в массиве возрастающую последовательность, начиная с первого элемента и перенести ее в другой массив (например, **3**,2,1,**4**,**6**,3,**8**,1,1,**12** -> 3,4,6,8,12).

Информатика-1. Практические занятия. Контрольная работа №3.

1. Анализ фрагмента, переменные и их "смысл", стандартные программные контексты

2. Содержательное описание алгоритма.

3. Оформление фрагмента в виде функции (заголовок, формальные параметры, результат, локальные переменные).

4. Вызов функции на статических (числовых) данных. Вывод результата через printf, численное значение результата.

```
//-----------------------------------------------------1
for ( int i=n1; !(n1 % i ==0 && n2 % i ==0); i--);
//-----------------------------------------------------2
for ( int n=a; n%a!=0 || n%b!=0; n++);
//-----------------------------------------------------3
for ( int n=2; n<a; n++)
     if (a%n==0) break;
if (n==a) return 1;
return 0;
//-----------------------------------------------------4
for ( int n=2; n<a && a%n!=0; n++);
if (n==a) return 1;
return 0;
//-----------------------------------------------------5
for ( int s=0,n=2; n<a; n++)
     if (a%n==0) s++;
if (s==0) return 1;
return 0;
//-----------------------------------------------------6
for ( int s=0,n=2; n<a; n++)
     if (a%n==0) { s=1; break; }
return s;
//-----------------------------------------------------7
int n=2; while(a%n!=0)
       {
       n++;
       if (n==a) return 1;
       }
return 0;
//-----------------------------------------------------8
for ( int i=0; i<n; i++)
     if (A[i]<0) break;
if (i==n) return 1;
return 0;
//-----------------------------------------------------9
for ( int i=0; i<n && A[i]>0; i++);
if (i==n) return 1;
return 0;
//-----------------------------------------------------10
```

```
for ( int s=0,i=0; i<n; i++)
     if (A[i]<0) s++;
if (s==0) return 1;
return 0;
//-----------------------------------------------------11
for ( int s=0,i=0; i<n; i++)
     if (A[i]<0) { s=1; break; }
return s;
//-----------------------------------------------------12
int i=0; while(A[i]>0)
       {
       i++;
       if (i==n) return 1;
       }
return 0;
//-----------------------------------------------------13
for (n=a, s=0; n!=0; n=n/10)
     { k=n%10; s=s+k;}
//-----------------------------------------------------14

for (n=a, s=0; n!=0; n=n/10)
     { k=n%10; if (k>s) s=k;}
//-----------------------------------------------------15
for (n=a, s=0; n!=0; n=n/10)
     { k=n%10; s=s*10+k;}
//-----------------------------------------------------16
for (i=0, n=a; n!=0; i++, n=n/10);
for (A[i--]=-1, n=a; n!=0; i--, n=n/10)
     A[i]=n % 10;
//-----------------------------------------------------17
for (j=0,a=10; a<30000; a++)
       {
       for (n=a, s=0; n!=0; n=n/10)
         { k=n%10; s=s+k;}
       if (a==s*s*s) A[j++]a;
//-----------------------------------------------------18
for (j=0,a=10; a<v; a++)
       {
       for (n=a, s=0; n!=0; n=n/10)
         { k=n%10; s=s*10+k;}
       if (a==s) A[j++]=a;
//-----------------------------------------------------19
for (i=0; i<n; i++)
       {
       if (A[i]<0){
              for (j=i; j<n-1;j++) A[j]=A[j+1];
              n--;
              i--;
              }
       }
//-----------------------------------------------------20
for (i=1,k=0; i<n; i++)
```

```
        if (A[i]>A[k]) k=i;

for (j=k; j<n-1;j++) A[j]=A[j+1];
        n--;
//-----------------------------------------------------21
for (i=0,a=2; a<v; a++)
        {
        for (n=2; n<a; n++)
          { if (a%n==0) break; }
        if (n==a) A[i++]=a;
        }
A[i]=0;
//-----------------------------------------------------22
for (i=0,a=2; a<v; a++)
        {
        for (s=0,n=2; n<a; n++)
                if (a%n==0) { s=1; break; }
        if (s==0) A[i++]=a;
        }
A[i]=0;
//-----------------------------------------------------23
for (j=0,i=0; i<n ; i++)
        {
        for (m=2; m<A[i]; m++)
          { if (A[i]%m==0) break; }
        if (m==A[i]) B[j++]=A[i];
        }
B[j]=0;
//-----------------------------------------------------24
for (j=0,i=0; i<n ; i++)
        {
        for (s=0,m=2; m<A[i]; m++)
          if (A[i]%m==0) { s=1; break; }
        if (s==0) B[j++]=A[i];
        }
B[j]=0;
//-----------------------------------------------------25
for (i=0; i<n ; i++)
        {
        for (m=2; m<A[i]; m++)
          { if (A[i]%m==0) break; }
        if (m==A[i]) {
                for (j=i; j<n-1;j++) A[j]=A[j+1];
                n--;
                i--;
                }
        }
//-----------------------------------------------------26
for (j=0,i=0; i<n ; i++)
        {
        for (s=0,m=2; m<A[i]; m++)
          if (A[i]%m==0) { s=1; break; }
```

```
        if (s==0) {
                for (j=i; j<n-1;j++) A[j]=A[j+1];
                n--;
                i--;
                }
        }
//-----------------------------------------------------27
for (i=0; i<n-1 && val !=1; i++)
        {
        for (m=2; val % m !=0; m++);
        val /= m; A[i] = m;
        }
A[i] = 0;
//-----------------------------------------------------28
v=A[0]+1;
do      {
        v--;
        for (i=0,s=0;i<n;i++)
                if (A[i]%v!=0) { s=1; break; }
        } while(s==1);
//-----------------------------------------------------29
for (i=0; i<n; i++)
        if (A[i]%v!=0) { v--; i=-1; }
//-----------------------------------------------------30
for (i=0,a=2; a<v && i<m-1 ; a++)
        {
        for (s=0,j=0; j<i; j++)
                if (a%A[j]==0) { s=1; break; }
        if (s==0) A[i++]=a;
        }
A[i]=0;
//-----------------------------------------------------31
for (i=0; i<n-1; i++)
for (j=i+1; j<n; j++)
          if (c[i]==c[j]) return i;
return -1;
//-----------------------------------------------------32
for (s=0,i=0; i<n; i++)
        {
        for (k=0,j=0; j<n; j++)
                if (c[i]==c[j]) k++;
        if (k>s) s=k,b=i;
        }
//-----------------------------------------------------33
for (s=0,i=0; i<n-1; i++)
    if (A[i]==A[i+1])
        {
        for (k=2; i+k<n && A[i]==A[i+k]; k++);
        if (k>s) s=k,b=i;
        }
//-----------------------------------------------------34
for (k=0, m=1; m <= n; k++, m = m * 2);
```

```
return k-1; }
//-----------------------------------------------------35
for (i=0,j=n-1; i < j; i++,j--)
     { k = c[i]; c[i] = c[j]; c[j] = k; }
//-----------------------------------------------------36
for (i=0; i<n; i++)
       {
       for (j=k1=k2=0; j<n; j++)
          if (c[i] != c[j])
               { if (c[i] < c[j]) k1++; else k2++; }
       if (k1 == k2) return i;
      }
return -1;
//-----------------------------------------------------37
for (s=0, i=0; i<n-1; i++)
       {
       for (j=i+1, m=0; j<n; j++)
       if (c[i]==c[j]) m++;
       if (m > s) s=m, b=i;
       }
//-----------------------------------------------------38
for (i=k=m=0; i<n-1; i++)
       if (c[i]==c[i+1]) k++;
       else
              {
              if (k > m) m=k, b=i-k-1;
              k=0;
              }
//-----------------------------------------------------39
for (s=0,i=0; i<n; i++)
       {
       if (A[i]<0) continue;
       if (A[i]==0) break;
       s=s+A[i];
       }
//-----------------------------------------------------40
for (s=0,i=0; i<n && A[i]>0; i++)
       s=s+A[i];
//-----------------------------------------------------41
for (k=0,s=0,i=0; i<n && k==0; i++)
       {
       if (A[i]<0) k=1;
       s=s+A[i];
       }
//-----------------------------------------------------42
for (s1=0,s2=0,i=0,j=n-1; i<=j;)
       {
       if (s1<s2) s1+=A[i],i++;
       else s2+=A[j],j--;
       }
return i;
//-----------------------------------------------------43
```

```
for (s=0,i=0; i<n; i++)
        {
        if (i%2==0)s=s+A[i];
        else s=s-A[i];
        }
//-----------------------------------------------------44
for(j=0;n!=0;j++)
        {
        for (k=0,i=1; i<n; i++)
                if (A[i]<A[k]) k=i;
        B[j]=A[k];
        for (;k<n-1;k++) A[k]=A[k+1];
        n--;
        }
//-----------------------------------------------------45
for(j=0,max=A[0];j<n;j++)
        if (A[j]>max) max=A[j];

for(j=0;j<n;j++)
        {
        for (k=0,i=1; i<n; i++)
                if (A[i]<A[k]) k=i;
        B[j]=A[k];
        A[k]=max+1;
        }
//-----------------------------------------------------46
while(n!=0)
        {
        for (k=0,i=1; i<n; i++)
                if (A[i]<A[k]) k=i;
        c=A[k]; A[k]=A[n-1]; A[n-1]=c;
        n--;
        }
//-----------------------------------------------------47
for (j=0,a=10; a<v; a++)
        {
        for (s=0,n=a, s=0; n!=0; n=n/10)
                {
          k=n%10;
          if (k!=0 && a%k!=0) { s=1; break; }
          }
        if (s==0) A[j++]=a;
        }
//-----------------------------------------------------48
for (i=0; i<n-1; i++)
    if (A[i]==A[i+1])
        {
        for (j=i; j<n-2;j++) A[j]=A[j+2];
        n=n-2;
        i--;
        }
//-----------------------------------------------------49
```

```
for (i=0,k=-1; i<10; i++)
       {
       if (A[i]<0) continue;
       if (k==-1) k=i;
       else
       if (A[i]<A[k]) k=i;
       }
//-----------------------------------------------------50
for (i=0,s=0,k=0; i<10; i++)
    if (A[i]<0) k=1;
    else
       {
       if (k==1) s++;
       k=0;
       }
//-----------------------------------------------------51
for (i=0,s=0; i<10; i++)
    if (A[i]>s) s=A[i];
//-----------------------------------------------------52
for (i=1,k=0; i<10; i++)
    if (A[i]>A[k]) k=i;
//-----------------------------------------------------53
for (i=0,k=-1; i<10; i++)
{ if (A[i]<0) continue;
if (k==-1) k=i;
else
    if (A[i]<A[k]) k=i;
}
//-----------------------------------------------------54
for (i=0,k=-1; i<10; i++)
{ if (A[i]<0) continue;
if (k==-1 || A[i]<A[k]) k=i;
}
//-----------------------------------------------------55
for (i=0,s=0; i<10; i++)
    if (A[i]>0) s++;
//-----------------------------------------------------56
for (i=1,s=0; i<10; i++)
    if (A[i]>0 && A[i-1]<0) s++;
//-----------------------------------------------------57
for (i=1,s=0,k=0; i<10; i++)
{
if (A[i-1]<A[i]) k++;
else
    { if (k>s) s=k;
    k=0;
    }
//-----------------------------------------------------58
for (i=0,s=0,k=0; i<10; i++)
    if (A[i]<0) k=1;
else
    { if (k==1) s++; k=0; }
```

```
//-----------------------------------------------------59
for (s=1, i=1; i<=n; i++) s = s * i;
//-----------------------------------------------------60
for (s=1, i=0; i<=n; i++) s = s * 2;
//-----------------------------------------------------61
for (s=0, i=0; i<n; i++) s = s + A[i];
//-----------------------------------------------------62
for (s=1, i=0; i<10; i++) s = s * A[i];
//-----------------------------------------------------63
for (n=2; n<a; n++)
    { if (a%n==0) break; }
if (n==a) puts("Good");
//-----------------------------------------------------64
for (s=0,n=2; n<a; n++)
    { if (a%n==0) s++; }
if (s==0) puts("Good");
//-----------------------------------------------------65
for (i=0; i<n; i++)
    if (A[i]<0) break;
if (i==n) puts("Good");
//-----------------------------------------------------66
for (i=2; i<a; i++)
    if (a%i==0) break;
if (i==a) puts("Good");
//-----------------------------------------------------67
for (s=A[0], i=1; i < 10; i++)
    A[i-1] = A[i];
A[9] = s;
```

Информатика-1. Практические занятия. Контрольная работа №4.

1. Анализ фрагмента, переменные и их "смысл", стандартные программные контексты

2. Содержательное описание алгоритма.

3. Оформление фрагмента в виде функции (заголовок, формальные параметры, результат, локальные переменные).

4. Вызов функции на статических (числовых) данных. Вывод результата через printf, численное значение результата.

```
//-----------------------------------------------------------1
for (i=0,nc=0;c[i]!=0;i++)
	if (c[i]!=' ' && (c[i+1]==' ' || c[i+1]==0)) nc++;
//-----------------------------------------------------------2
for (i=0,nc=0;c[i]!=0;i++)
	if (c[i]!=' ' && (i==0 || c[i-1]==' ')) nc++;
//-----------------------------------------------------------3
for (i=0;c[i]!=0;i++);
c[i]='*'; c[i+1]=0;
//-----------------------------------------------------------4
for (i=0;c[i]!=0;i++)
	if (c[i]==' ' && c[i+1]==' ')
		{
		for (k=i;c[k]!=0;k++) c[k]=c[k+1];
		i--;
		}
//-----------------------------------------------------------5
for (i=0,j=0;c[i]!=0;i++)
	if (c[i]!=' '){
		out[j++]=c[i];
		if (c[i+1]==' ') out[j++]=' ';
		}
out[j]=' ';
//-----------------------------------------------------------6
for (i=0,j=0;c[i]!=0;i++)
	if (!(c[i]==' ' && c[i+1]==' '))
		out[j++]=c[i];
out[j]=0;
//-----------------------------------------------------------7
for (i=0,nc=0;c[i]!=0;i++)
	if (c[i]>='0' && c[i]<='9')
		nc=nc+c[i]-'0';
//-----------------------------------------------------------8
for (i=0,nc=0;c[i]!=0;i++)
	if (c[i]!=' ' && (i==0 || c[i-1]==' '))
		B[nc++]=i;
//-----------------------------------------------------------9
for (i=0,j=0;c[i]!=0;i++)
	if (c[i]!=' ' && (c[i+1]==' ' || c[i+1]==0))
		{
```

```
            for (k=i; k>=0 && c[k]!=' ';k--)
                  out[j++]=c[k];
            out[j++]=' ';
            }
out[j]=0;
//-----------------------------------------------------------10
for (i=0;c[i]!=0;i++)
      if (c[i]>='0' && (c[i]<='9') break;
for (s=0;c[i]>='0' && c[i]<='9';i++)
       s=s*10+c[i]-'0';
//-----------------------------------------------------------11
k=0,b=-1,m=0;
for (i=0;c[i]!=0;i++)
      if (c[i]==c[i+1])
            {
            for (k=2;c[i]==c[i+k];k++);
            if (k>m) m=k,b=i;
            }
//-----------------------------------------------------------12
k=0,b=-1,m=0;
for (i=0;c[i]!=0;i++)
      if (c[i]!=' ') k++;
      else
            {
            if (k!=0 && k>m) m=k,b=i-k;
            k=0;
            }
//-----------------------------------------------------------13
k=0,b=-1,m=0;
for (i=0;c[i]!=0;i++)
      if (c[i]!=' '){
            for (k=1;c[i+k]!=' ' && c[i+k]!=0;k++);
            if (k>m) m=k,b=i;
            i=i+k-1;
            }
//-----------------------------------------------------------14
for (i=0;c[i]!=0;i++)
      if (c[i]>='0' && c[i]<='9' || c[i]>='A' && c[i]<='F') break;
for (s=0;c[i]>='0' && (c[i]<='9' || c[i]>='A' && c[i]<='F';i++)
       if (c[i]>='0' && (c[i]<='9')
             s=s*16+c[i]-'0';
       else
             s=s*16+c[i]-'A'+10;
//-----------------------------------------------------------15
for (i=0; c[i] !='\0'; i++);
for (j=0,i--; i>j; i--,j++)
       { char s; s=c[i]; c[i]=c[j]; c[j]=s; }}
//-----------------------------------------------------------16
for (i=0, old=0, nw=0; c[i] !='\0'; i++)
      {
      if (c[i]==' ') old = 0;
      else { if (old==0) nw++; old=1; }
```

```
	}
//-----------------------------------------------------------17
for (mm=a, k=0; mm !=0; mm /=10 ,k++);
for (mm=a, c[k--]='\0'; k>=0; k--,mm=mm/10)
	c[k]= mm % 10 + '0';
//-----------------------------------------------------------18
for (mm=a, k=0; mm !=0; mm /=16 ,k++);
for (mm=a, c[k--]='\0'; k>=0; k--,mm=mm/16)
	{
	int v=mm % 16;
	if (v <=9) c[k]= v + '0';
	else c[k]= v - 10 + 'A';
	}
//-----------------------------------------------------------19
for (n=0,ns=0; c[n] !='\0'; n++)
	{
	for (k=0; n-k >=0 && c[n+k] !='\0'; k++)
		if (c[n-k] != c[n+k]) break;
	if (k >=3) ns++;
	}
//-----------------------------------------------------------20
for (i=0; c1[i] !='\0'; i++)
	{
	for (j=0; c1[i+j]==c2[j]; j++)
		if (c2[j]=='\0') return i;
	}
//-----------------------------------------------------------21
for (z='?',s=0,m='A'; m <='Z'; m++)
	{
	for (n=0, i=0; c[i] !='\0'; i++)
		if (c[i]==m) n++;
	if (n > s) { z=m; s=n; }
	}
//-----------------------------------------------------------22
double x;
int i;
x-=(int)x;
for (c[0]='.', x -= (int)x, i=1; i<6; i++)
	{
	x *= 10.; c[i]=(int)x + '0'; x -= (int)x;
	}
c[i]='\0';
//-----------------------------------------------------------23
for (i=j=cm=0; c[i] !='\0'; i++)
	{
	if (c[i]=='*' && c[ i+1]=='/') { cm--, i++; continue; }
	if (c[i]=='/' && c[ i+1]=='*') { cm++, i++; continue; }
	if (cm==0) out[j++] = c[i];
	}
out[j]=0;
```