Учреждение образования
«Гомельский государственный университет имени Франциска Скорины»

Факультет математики и технологий программирования

Кафедра математических проблем управления и информатики

СОГЛАСОВАНО
Заведующий кафедрой
математических проблем
управления и информатики

_____ В.С. Смородин
22.11 _____ 2022 г.

СОГЛАСОВАНО
Декан
факультета математики и
технологий программирования

_____ Е.П. Жогаль
9.12 _____ 2022 г.

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

ПРОГРАММИРОВАНИЕ
(часть 2)

для специальности
1-31 03 07-01 Прикладная информатика
(программное обеспечение компьютерных систем)

Составители:
Короткевич В.А., доцент кафедры математических проблем управления и информатики, к.т.н., доцент,
Короткевич Л.И., старший преподаватель кафедры математических проблем управления и информатики

Рассмотрено на заседании
кафедры математических проблем управления и информатики
22.11 2022г., протокол № 4.

Рассмотрено и утверждено
на заседании научно-методического совета университета
23.12 2022г., протокол № 5.

Гомель, 2022

СОДЕРЖАНИЕ УЧЕБНО-МЕТОДИЧЕСКОГО КОМПЛЕКСА ПО ДИСЦИПЛИНЕ «ПРОГРАММИРОВАНИЕ» (часть 2)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к учебно-методическому комплексу
по дисциплине компонента учреждения высшего образования
«Программирование» (часть 2)
для специальности
1-31 03 07-01 Прикладная информатика
(программное обеспечение компьютерных систем)

Учебно-методический комплекс по дисциплине компонента учреждения высшего образования «Программирование» (часть 2) составлен в соответствии с учебным планом учреждения образования для специальности 1-31 03 07-01 Прикладная информатика (программное обеспечение компьютерных систем), утвержденным 15.04.2020 (регистрационный № G 31-01-20/УП) и учебной программой учреждения высшего образования, утвержденной 20.05.2020 (регистрационный № УД-12-2020-29 /уч.). Данные документы предусматривают формирование у студентов устойчивых базовых теоретических знаний и практических навыков в области программирования, овладение студентами методами и приемами разработки программных приложения на различных языках программирования, а также техникой тестирования и отладки приложений. В них заложена задача формирования у будущих специалистов профессиональных и личностных компетенций в области программирования. Изучение данной дисциплины является необходимым этапом в профессиональном развитии специалиста в области информационных технологий и позволяет в дальнейшем совершенствовать навыки разработки профессиональных программных средств, отвечающих современному этапу развития компьютерной техники.

Данный учебно-методический комплекс предназначен для информационно-методического обеспечения преподавания дисциплины компонента учреждения высшего образования «Программирование».

Целью учебно-методического комплекса по дисциплине «Программирование» (часть 2) является обеспечение теоретической и практической подготовки студентов, активизации их учебно-познавательной деятельности по вопросам проектирования эффективных алгоритмов, выбора наиболее подходящих программных и технических средств их реализации, разработки программных приложений на языке С, отвечающих современным требованиям, развитие профессионально-личностных компетенций в данной сфере, совершенствование умений и навыков самостоятельной учебной работы.

Задачами учебно-методического комплекса являются:

– раскрыть требования к содержанию учебной дисциплины «Программирование»;

– обеспечить успешное усвоение теоретического материала по дисциплине;

– актуализировать использование традиционных форм и методов контроля знаний и стимулировать инновационные подходы к проверке и оценке знаний, умений и навыков студентов;

– повысить уровень творческой активности студентов через выполнение различных видов творческих заданий и проблемно-поисковых задач.

В структурном отношении учебно-методический комплекс по дисциплине «Программирование» (часть 2) включает в себя четыре раздела: теоретический, практический, раздел контроля знаний, вспомогательный.

Теоретический раздел содержит основные положения, выносимые на лекции, и включает в себя 20 тем, предназначенных для аудиторной работы со студентами (лекции преподавателя – 52 часа). В результате изучения этих тем студенты получат базовые теоретические знания в области программирования на языке С, в построении эффективных алгоритмов решения поставленных задач, в выборе наиболее подходящих структур данных, программных и технических средств реализации алгоритмов. Также студенты получат основные теоретические сведения по организации эффективной отладки и тестированию прикладного программного обеспечения.

Материал, представленный в теоретическом разделе УМК, поможет студентам изучить основные этапы разработки программ на языке программирования С, базовые конструкции языка, методы и средства управления ресурсами компьютера. Комплекс содержит теоретические сведения, позволяющие разрабатывать программные приложения на языке С для решения различных практических задач, а также практические рекомендации по разработке и реализации эффективных алгоритмов, выбору подходящих программных средств для их реализации.

Эффективное использование комплекса для изучения теоретического материала предполагает организацию самостоятельной работы студентов перед лекциями. На лекциях целесообразно не столько излагать тот или иной учебный материал УМК, сколько обсуждать его, акцентируя внимание на принципиально важных и сложных учебных фрагментах. Для такой самостоятельной работы студентов предназначены краткий конспект и презентации лекций, входящие в теоретический раздел комплекса.

При изучении дисциплины «Программирование» очень важны те навыки и умения, которые студенты должны получить в ходе лабораторных занятий.

Практический раздел УМК включает в себя в соответствии с учебным планом дисциплины 42 темы лабораторных занятий. Также практический раздел включает в себя лабораторный практикум для проведения лабораторных занятий, который структурирован по темам. Все лабораторные работы имеют 25 вариантов, что позволяет выдавать индивидуальные задания всем студентам группы. Данный лабораторный практикум охватывает практически все изучаемые теоретические темы и позволит студентам овладеть методами и приемами разработки эффективных алгоритмов решения задач, приобрести навыки в реализации этих алгоритмов на языке С, а также навыки по отладке и тестированию разработанных приложений. Практические навыки, приобретенные в ходе выполнения заданий лабораторных работ, позволят студентам овладеть методами и приемами разработки современных и эффективных приложений на языке С.

Включенные в практический раздел примеры решения задач лабораторных работ позволят студентам быстрее решать собственные задания из лабораторного практикума, а также разрабатывать оптимальные алгоритмы решения задач и использовать более эффективные средства языка С при их реализации. Входящее в УМК практическое руководство поможет студентам качественно выполнить лабораторные работы.

Раздел контроля знаний учебно-методического комплекса по дисциплине «Программирование» (часть 2) включает в себя перечни вопросов к зачету и экзамену, которые охватывают весь материал, изучаемый студентами, а также примеры экзаменационных задач, которые позволят студентам самостоятельно и качественно подготовится к экзаменам.

Использование электронных тестов из раздела контроля знаний комплекса позволит студентам эффективно подготовиться к лабораторным занятиям, контрольным работам, зачету и экзамену. Тесты, включенные в УМК, носят вспомогательный характер. Это связано с тем, что в ходе изучения дисциплины студенты должны, прежде всего, приобрести навыки решения практических задач и основное внимание следует уделить выполнению студентами заданий лабораторных работ.

Вспомогательный раздел учебно-методического комплекса содержит необходимые элементы учебно-программной документации: учебную программу учреждения образования по дисциплине «Программирование» с пояснительной запиской и содержанием учебного материала. Кроме того, в данном разделе имеется перечень рекомендуемой для студентов литературы.

При работе с УМК по дисциплине «Программирование» (часть 2) предполагается использование традиционных способов коллективного обучения: лекций, лабораторных занятий, индивидуальных заданий с последующей отчетностью. Применяемые информационные технологии: лекции в форме презентаций и тестирующие программы.

Знания, умения и навыки, приобретенные в ходе изучения материала учебно-методического комплекса, позволят студентам разрабатывать эффективные программные приложения для решения учебных, научных и прикладных задач.

Учебно-методический комплекс по учебной дисциплине «Программирование» (часть 2) адресуется студентам 1-го курса дневной формы обучения специальности 1-31 03 07-01 Прикладная информатика (программное обеспечение компьютерных систем).

1.1. ПЕРЕЧЕНЬ ТЕОРЕТИЧЕСКОГО МАТЕРИАЛА ПО ДИСЦИПЛИНЕ «ПРОГРАММИРОВАНИЕ» (часть 2)

Раздел 5. Язык программирования С

Тема 5.1. Базовые элементы языка программирования С

Основные этапы разработки программ на языке С. Алфавит языка, ключевые слова, идентификаторы, операторы, инструкции, разделяющие знаки. Переменные и константы. Комментарии и их использование.

Типы данных. Встроенные и пользовательские типы данных. Простые и структурированные типы данных. Целочисленные типы данных. Представление в памяти целочисленных типов данных. Вещественные типы данных. Элементарный ввод-вывод.

Тема 5.2. Операции и выражения

Понятия операции и выражения. Операция присваивания. Множественное присваивание. Арифметические операции. Операции сравнения и логические операции. Приоритеты операций в выражениях. Преобразование типов в выражениях. Явное приведение типов.

Тема 5.3. Операторы управления вычислительным процессом

Реализация базовых алгоритмических структур средствами языка программирования – управляющие инструкции. Условный оператор if…else и if. Оператор множественного выбора switch.

Цикл while: особенности и примеры использования. Цикл do…while. Цикл со счетчиком for. Бесконечные циклы. Инструкции break и continue.

Тема 5.4. Адресная арифметика

Понятие указателя. Объявление указателя. Операторы взятия адреса и разыменования. Присваивание указателей. Нетипизированные указатели. Неявное и явное приведение типов указателей. Арифметические операции с указателями. Сравнение указателей. Указатели на указатели.

Тема 5.5. Массивы

Определение массива. Одномерные массивы. Размещение одномерных массивов в памяти. Объявление и инициализация массива. Двумерные и многомерные массивы. Доступ к элементам массивов: оператор индексирования.

Указатели на массивы. Доступ к элементам массивов с помощью указателей. Массивы указателей.

Базовые алгоритмы обработки массивов. Двоичный поиск.

Тема 5.6. Строки символов

Понятие строки в языке С. Объявление и инициализация строк. Указатели на строки и символы. Инициализация указателей на символы с помощью строковых констант. Доступ к элементам строки по индексу. Доступ к эле-

ментам строки с помощью указателя. Массивы строк.

Базовые алгоритмы обработки строк.

Тема 5.7. Функции

Определение функции. Объявление функции. Имя и тело функции. Параметры функции. Прототип функции. Механизмы передачи параметров в вызываемую функцию и из вызываемой функции. Передача параметров по значению.

Тема 5.8. Классы хранения и видимость переменных

Общие положения. Область определения и видимость идентификатора. Локальные и глобальные переменные, их области видимости. Автоматические переменные. Статические переменные. Класс хранения static.

Тема 5.9. Разработка и использование функций

Передача указателей в функции. Инициализация параметров функции. Возвращаемые значения. Возврат указателей. Функции, не возвращающие значения. Передача параметров в функцию main(). Примеры разработки и использования функций.

Тема 5.10. Рекурсивные функции

Виды рекурсии. Понятие рекурсивной функции. Примеры рекурсивных функций. Основные преимущества и недостатки применения рекурсивных функций. Правила разработки рекурсивных функций. Типичные ошибки при разработке рекурсивных функций. Отладка рекурсивных функций.

Тема 5.11. Библиотечные функции обработки и преобразования данных

Понятие библиотеки функций. Функции преобразования данных. Стандартные математические функции. Функции классификации и преобразования символов. Функции для работы с блоками памяти. Функции для работы со строками символов. Примеры работы с библиотечными функциями.

Тема 5.12. Работа с битами

Основные понятия. Двоичная система счисления. Побитовые операции. Типичные операции при работе с битами. Битовые маски. Примеры программ для работы с битами.

Тема 5.13. Структуры, объединения и перечисления

Общие положения. Шаблон структуры. Внешний и внутренний шаблоны. Структурные переменные. Правила выравнивания структурных переменных в памяти. Оператор typedef описания собственного типа данных. Вложенные структуры. Указатели на структуры. Доступ к членам структур с помощью указателей. Массивы структурных переменных. Использование структур в функциях. Объединения. Перечисления.

Тема 5.14. Препроцессор

Общие положения. Директивы #define и #undef. Обработка директив #define и #undef. Включение файлов. Директива #include. Условная компиля-

ция. Директивы #if, #else, #elif.

Раздел 6. Управление ресурсами в языке С

Тема 6.1. Управление памятью

Статическое и динамическое распределение памяти. Функции динамического выделения и освобождения памяти. Оператор определения размера (sizeof). Динамическое выделение памяти для массивов и структур. Примеры на выделение и освобождение памяти.

Тема 6.2. Работа с файлами

Общие положения. Режимы доступа к файлам. Открытие и закрытие файла. Режимы открытия файлов. Доступ к файлам через поток ввода-вывода. Понятие потока ввода-вывода. Посимвольный и построчный ввод вывод. Форматированный ввод-вывод.

Произвольный доступ к файлам. Блоковый ввод-вывод информации. Префиксный доступ к файлам. Функции префиксного файлового ввода-вывода. Управление указателем чтения-записи.

Тема 6.3. Связанные динамические структуры данных

Понятие динамической структуры данных. Структуры, ссылающиеся на себя. Связанные списки. Стеки. Очереди. Выделение и освобождение памяти под элементы динамических структур данных. Реализация динамических структур данных. Удаление динамических структур данных.

Тема 6.4. Списки, виды списков

Виды списков. Однонаправленные списки. Кольцевые списки. Двунаправленные списки. Реализация списков. Методы доступа к элементам списка. Удаление списка.

Тема 6.5. Обработка списков

Стандартные процедуры обработки списков: просмотр списка, поиск в списке, вставка нового элемента в список, удаление элемента из списка, сортировка списка.

Тема 6.6. Управление файлами, директориями и накопителями

Создание и уничтожение файла, директория. Управление текущим накопителем и директорием. Чтение содержимого директория. Поиск файлов. Удаление и переименование файлов. Создание резервных копий файлов. Определение существования файла или директория. Определение и установка параметров файла.

2.1. ПЕРЕЧЕНЬ ЛАБОРАТОРНЫХ ЗАНЯТИЙ ПО ДИСЦИПЛИНЕ «ПРОГРАММИРОВАНИЕ» (часть 2)

1. Типы данных
2. Элементарный ввод-вывод
3. Операции и выражения
4. Условный оператор
5. Циклы
6. Указатели
7. Арифметические операции с указателями
8. Массивы
9. Доступ к элементам массивов: оператор индексирования
10. Доступ к элементам массивов с помощью указателей
11. Базовые алгоритмы обработки массивов
12. Строки
13. Указатели на строки и символы
14. Базовые алгоритмы обработки строк
15. Функции
16. Механизмы передачи параметров
17. Классы хранения и видимость переменных
18. Передача указателей в функции
19. Передача параметров в функцию main()
20. Примеры разработки и использования функций
21. Рекурсивные функции
22. Библиотечные функции обработки и преобразования данных
23. Функции для работы со строками символов
24. Работа с битами
25. Типичные операции при работе с битами
26. Структуры, объединения и перечисления
27. Препроцессор
28. Управление памятью
29. Динамическое выделение памяти для массивов и структур
30. Работа с файлами
31. Доступ к файлам через поток ввода-вывода
32. Форматированный ввод-вывод
33. Префиксный доступ к файлам
34. Связанные динамические структуры данных
35. Списки, виды списков
36. Реализация списков
37. Методы доступа к элементам списка
38. Поиск в списках
39. Вставка и удаление элементов списка

40. Сортировка списков
41. Управление файлами, директориями и накопителями
42. Чтение содержимого директория

2.2. МАТЕРИАЛЫ (ЗАДАНИЯ) ДЛЯ ПРОВЕДЕНИЯ ЛАБОРАТОРНЫХ ЗАНЯТИЙ ПО ДИСЦИПЛИНЕ «ПРОГРАММИРОВАНИЕ» (часть 2)

ЛАБОРАТОРНАЯ РАБОТА №1

«Типы данных, ввод/вывод, операторы управления вычислительным процессом»

Условия и ограничения для работы №1

1. Исходные данные в задачах 1.1 и 1.2 имеют тип int.
2. Исходные данные в задаче 1.3 имеют тип unsigned long.
3. Массивы использовать нельзя.
4. Задачи 1.2 и 1.3 выполнить, используя цикл while.
5. В задаче 1.2 число, служащее признаком окончания ввода, не является числом последовательности.
6. В задачах 1.1, 1.2 и 1.3 математические функции использовать нельзя.
7. Условие задачи 1.4. Используя цикл for, вывести таблицу значений функции f(x) с параметром u с точностью до четырех знаков после точки в n равноудаленных и равномерно распределенных точках на отрезке [a; b]. Обеспечить контроль корректности как исходных, так и промежуточных данных.
8. В задаче 1.4 обеспечить вывод номера ветки (от 1 до 3), по которой вычисляется значение функции.

Вариант №1

1. Пусть на прямой заданы точка k и отрезок [x; y]. Найти расстояние от точки до ближайшего конца отрезка.
2. Найти положительный минимум из чисел последовательности. Окончание ввода – число 0.
3. Даны два числа. Образовать новое число, оставив k младших цифр второго числа, где k – старшая цифра первого числа (362, 21589 => 589).
4. $$f(x)=\begin{cases}\ln(x/(u-5\cdot x)), |x|<10\\ \sqrt{1/u\cdot u/x}, x>30\\ x^4\cdot\sin(x), \text{в остальных случаях}\end{cases}$$

Вариант №2

1. Для трех чисел найти номер первого нулевого из них. Если числа 0 нет, выдать соответствующее сообщение.
2. Найти разность между количеством четных и нечетных чисел последовательности, не принадлежащих интервалу [-5; 5]. Окончание ввода – число 1000.
3. Дважды переписать число в обратном порядке (3081 => 18031803).
4. $$f(x)=\begin{cases}e^{-x}/\sqrt{u\cdot x^4}, x<-5\\ \cos(x+\ln(x)), x>30 \text{ или } x\in[8;20]\\ 1-e^x+x^6, \text{в остальных случаях}\end{cases}$$

Вариант №3

1. Для трех чисел x, y и z найти, сколько из них не попадает в интервал [a; b]. Обеспечить контроль

ввода границ интервала: если границы заданы неверно (a>b), выдать соответствующее сообщение.
2. Найти, сколько раз в последовательности чисел идут подряд два нулевых числа. Окончание ввода – отрицательное число.
3. Вставить после каждой цифры 3 цифру 0 в записи числа (25303 => 2530030).
4. $f(x)=\begin{cases}(x^5-u)/\ln(x), x\in[1;20]\\ x\cdot\cos(x), |x|>30\\ 3+\sqrt[4]{x+u}, \text{в остальных случаях}\end{cases}$

Вариант №4

1. Если для трех чисел сумма двух первых больше произведения двух последних, найти квадрат произведения всех трех чисел, в противном случае – найти утроенное второе число.
2. Найти сумму двузначных чисел последовательности, кратных 3. Окончание ввода – число 0.
3. Даны два числа. Добавить к первому числу перевернутое второе число (1457, 231 => 1457132).
4. $f(x)=\begin{cases}\sin(x+e^u), x>10\\ \ln(x+u), |x|<3 \text{ или } x\in[-6;-4]\\ -x^3/(u\cdot x+100), \text{в остальных случаях}\end{cases}$

Вариант №5

1. Ввести четыре числа и найти максимум из первых двух и минимум из последних двух. Выдать сообщение, что больше – найденный максимум или найденный минимум.
2. Найти количество чисел последовательности, расположенных между первым и вторым числами, равными m. При отсутствии двух таких чисел вывести -1. Окончание ввода – число 0.
3. Найти количество пар соседних повторяющихся цифр в записи числа (1445777 => 3).
4. $f(x)=\begin{cases}x|x+u|, -5<x<4\\ \ln(\cos^2(x)), |x|>20\\ e^{-x}/(x-10), \text{в остальных случаях}\end{cases}$

Вариант №6

1. Если сумма трех чисел положительная, найти, сколько среди этих трех чисел положительных, в противном случае – найти, сколько среди них нулей.
2. Найти наибольшее количество подряд идущих однозначных чисел последовательности. Окончание ввода – число 500.
3. Оставить в записи числа первую и последнюю цифры (2336027 => 27).
4. $f(x)=\begin{cases}\ln(u\cdot x^2-4), 2<x<10\\ (x+u)/(x+25)^2, |x|>20\\ e^{-x}/x, \text{в остальных случаях}\end{cases}$

Вариант №7

1. Определить, можно ли из четырех длин сторон, введенных в порядке возрастания, построить прямоугольник.
2. Для последовательности чисел, найти номер первого числа в последней паре идущих подряд значений, равных k. Окончание ввода – число 0.
3. Удалить справа подряд идущие нечётные цифры в записи числа и добавить перед первой цифрой остаток от деления на 10 количества удалённых нечётных цифр (25187513 => 42518).
4. $f(x)=\begin{cases}\sqrt{u-x^2}, |x|<3\\ e^x\cdot x^6, x\in[-10;-5]\\ \sin(\ln(|x|)), \text{в остальных случаях}\end{cases}$

Вариант №8

1. Для трех чисел найти среднее арифметическое тех из них, которые попадают в интервал [n; m]. Если ни одно из чисел не попадает в интервал [n; m], выдать соответствующее сообщение.
2. Найти общее количество однозначных и трехзначных чисел последовательности из заданного интервала [a; b]. Окончание ввода – число, кратное 5.
3. Даны два числа. Определить, входит ли первое число во второе при их написании (145, 231456 => да).
4. $$f(x)=\begin{cases}\sqrt{u\cdot x}, -5<x<20\\ \cos^{-2}(\pi\cdot(x+u)), |x|>30\\ x^3+x^2-15, \text{в остальных случаях}\end{cases}$$

Вариант №9

1. Найти расстояние от точки x до отрезка [a; b] на прямой. Если точка принадлежит отрезку, считать, что расстояние равно 0.
2. Для последовательности чисел найти сумму чисел в наиболее короткой подпоследовательности, состоящей из двузначных чисел. Окончание ввода – число 0.
3. Дано число. Начиная от младшего разряда, получить новое число путем выбора младшего разряда от произведения каждой пары цифр исходного числа (13463755 => 3415).
4. $$f(x)=\begin{cases}u/(x-3)^2, -2<x<4\\ \sqrt{x+u/(x-10)^3}, x\in[8;20]\\ \sin(e^x), \text{в остальных случаях}\end{cases}$$

Вариант №10

1. Если для двух чисел хотя бы одно из чисел равно 0, выдать соответствующее сообщение. В противном случае – найти их разность, если числа имеют один знак, или их сумму, если числа имеют разные знаки.
2. Найти общее количество чисел последовательности до первого нулевого элемента и после последнего нулевого элемента. Окончание ввода – не равное 0 число, кратное 4.
3. Циклически сдвинуть цифры в записи числа на один разряд вправо (25806 => 62580).
4. $$f(x)=\begin{cases}\sqrt{u\cdot x}, |x|<4\\ \sin(x+u), x>30\\ e^{-x}/(u+x^2), \text{в остальных случаях}\end{cases}$$

Вариант №11

1. Найти модуль среднего арифметического трех чисел.
2. Найти номера первого отрицательного и последнего положительного чисел последовательности (если в последовательности нет отрицательных или положительных чисел, выдавать в качестве соответствующего номера -1). Окончание ввода – число 0.
3. Даны два числа. Получить новое число, дважды сцепив запись второго числа с первым (150, 344 => 344150344).
4. $$f(x)=\begin{cases}\sqrt{u\cdot x}\cdot\sin(x), 0\le x\le 10\\ x-1+1/(x-u), x>10\\ \ln(-u\cdot x+5), \text{в остальных случаях}\end{cases}$$

Вариант №12

1. Для трех чисел найти наименьшее целое число, которое больше всех этих чисел.
2. Найти сумму чисел последовательности между первым нечетным и первым нулевым элементами последовательности. Окончание ввода – число 500 или отрицательное число.
3. Начиная от младшего разряда, удалить каждую третью цифру числа (1234567 => 13467).

4. $$f(x)=\begin{cases}5\cdot\sin(x+1)/u, x\le 0\\ \sqrt[3]{x/5}-1+1/(u\cdot x-3), x>10\\ -x^2\cdot\ln(|x-10|), \text{в остальных случаях}\end{cases}$$

Вариант №13

1. Для трех чисел и интервала [a; b] определить, сколько из них не попадает в интервал, попадает внутрь интервала, попадает на границы интервала.
2. Определить, является ли последовательность чисел арифметической прогрессией. Окончание ввода – число, кратное 10.
3. Получить новое число, в котором сначала идут четные, а затем нечетные цифры исходного числа (3245349 => 2443539).
4. $$f(x)=\begin{cases}\sqrt{|x-1|}\cdot\sin(x^2-10), 0\le x\le 5\\ -x^3+x/(\ln(u\cdot x)-1), x>15\\ (x+5)/tg(\pi\cdot x), \text{в остальных случаях}\end{cases}$$

Вариант №14

1. Если оба введенных числа отрицательные, найти их среднее арифметическое. Иначе, если первое число больше второго, найти их сумму, в противном случае – найти их произведение.
2. Найти произведение чисел последовательности, кратных 3. Окончание ввода – модуль произведения стал больше 10000.
3. Начиная от младшего разряда, поменять местами соседние цифры числа (56234 => 502643).
4. $$f(x)=\begin{cases}\sin(x)\cdot(x+e), x\le 1\\ 3^x+\sqrt[3]{x/2}-1/u, x>10\\ 1/\ln(u\cdot x-5), \text{в остальных случаях}\end{cases}$$

Вариант №15

1. Пусть на прямой заданы два отрезка [a; b] и [c; d]. Найти длину их пересечения.
2. Найти максимальную разность между соседними числами последовательности. Окончание ввода – число не из интервала [-1000; 1000].
3. Уменьшить каждую цифру числа на наименьшую цифру в его записи (255289 => 33067).
4. $$f(x)=\begin{cases}\sqrt{u\cdot x}\cdot 2^{x+1}\cdot\cos(x), x\le 0\\ -x/\sqrt{|x-5\cdot u|}, 0<x<4\\ e^x\cdot\ln(x^2-25), \text{в остальных случаях}\end{cases}$$

Вариант №16

1. Найти минимальное число из четырех чисел.
2. Найти сумму чисел последовательности с нечетными номерами, предшествующих первому элементу, равному k. Окончание ввода – число 0 или число, меньшее 100.
3. Удалить из числа все нечетные цифры, добавив в конец числа вместо каждой из них цифру 0 (567245 => 624000).
4. $$f(x)=\begin{cases}\sqrt{u\cdot x^2-2}, -2<x<10\\ \cos(x), x<-7\\ \ln(u+x)/e, \text{в остальных случаях}\end{cases}$$

Вариант №17

1. Найти наибольшее целое отрицательное число, которое меньше всех трех введенных чисел.
2. Найти количество чисел последовательности, находящихся за последним числом, равным n.

Окончание ввода – число из интервала [100; 200].

3. От цифр на четных позициях отнять 1, причем 0–1=9 (позиции нумеруются справа налево по степени десятки, т.е. младший разряд нулевой, следующий разряд первый и т.д.) (52076 => 42975).
4. $$f(x)=\begin{cases}\sqrt[3]{x^2}, x\le 0\\ \ln(x-u)\cdot|5-x|, x\ge 5\\ \sin^2(x)/(x-1), \text{в остальных случаях}\end{cases}$$

Вариант №18

1. Если для чисел a и b выполняется условие a < b, ввести третье число d и определить, попадает ли d в интервал [a; b]. Если же a ≥ b, ввести еще два числа x и y и найти максимальное из них.
2. Для последовательности чисел найти количество чисел в наиболее длинной подпоследовательности из одинаковых чисел. Окончание ввода – число 0.
3. Присоединить в конец первому числу две первых, а в начало одну последнюю цифры второго числа (345, 75649 => 934575).
4. $$f(x)=\begin{cases}\sqrt{|x-3|+u}/(x+1), 0\le x\le 3\\ e^x+1/(u\cdot x^2), x<0\\ 3^{x+2}\cdot\sin(x-3), x>3\end{cases}$$

Вариант №19

1. Для двух отрезков [a; b] и [x; y] на прямой определить их взаимное расположение: отрезки не пересекаются, отрезки пересекаются, один отрезок находится внутри другого.
2. Определить количество чисел последовательности, делителем которых является ее первый член. Окончание ввода – отрицательное число.
3. Удалить из первого числа все цифры, равные минимальной цифре второго числа (217160, 351 => 2760).
4. $$f(x)=\begin{cases}\sqrt{x^3}/u^2, x<5\\ \ln(3-u\cdot x)^{-1}, x\in[5;10]\\ \sin(e^x), \text{в остальных случаях}\end{cases}$$

Вариант №20

1. Ввести четыре числа. Если сумма последних трех чисел больше первого числа, найти удвоенное первое число, в противном случае – найти максимальное среди второго и третьего числа.
2. Найти номер того числа последовательности, которое имеет максимальную сумму предыдущего и последующего чисел. Окончание ввода – число 0.
3. Обнулить все цифры в числе после первой четной цифры (3163850 => 3160000).
4. $$f(x)=\begin{cases}2^{x+5}\cdot\sin(x), x\le 0\\ -e^x/\sqrt{|u\cdot x-5|}, 0<x<4\\ \sqrt{x}\cdot\ln(x^2-81), x\ge 4\end{cases}$$

Вариант №21

1. Среди трех чисел найти среднее число (не максимальное и не минимальное). Если такого числа нет, выдать соответствующее сообщение.
2. Найти количество четных чисел последовательности, превышающих предыдущее число не менее чем на 5. Окончание ввода – отрицательное число кратное 3.
3. Удалить из записи числа все цифры от 2 до 5 (163825 => 168).

4. $f(x)=\begin{cases} x^5/(x^2-u), |x|<16 \\ \ln(\cos(x)), x>30 \\ e^{-\sin(x)}, \text{в остальных случаях} \end{cases}$

Вариант №22

1. Для трех чисел найти их произведение без максимального элемента.
2. Найти количество чисел последовательности, чей модуль совпадает с первым числом последовательности. Окончание ввода – числа 0 или 1000.
3. Начиная делить число на пары цифр от младшего разряда, получить новое число путем выбора наименьшей цифры из каждой пары цифр исходного числа (163855 => 135).

4. $f(x)=\begin{cases} \sqrt{7}/(u\cdot x-5), 6<x<12 \\ \sqrt{x/u}\cdot e^x, x<3 \\ \ln|\sin(\frac{\pi}{2}x)|, \text{в остальных случаях} \end{cases}$

Вариант №23

1. Найти сумму наибольшего и наименьшего из трех чисел.
2. Найти количество чисел последовательности, чей модуль совпадает с номером числа в последовательности. Окончание ввода – число 1000 или число -1000.
3. Получить новое число путем попарного сцепления цифр одного разряда двух чисел (1491, 24 => 10409214).

4. $f(x)=\begin{cases} x^3/(u+x), x\in[1;20] \\ e^{|x|/300}, |x|>30 \\ \sqrt[4]{\ln(x)+u}, \text{в остальных случаях} \end{cases}$

Вариант №24

1. Для трех чисел, если все три числа равны 0, выдать соответствующее сообщение, в противном случае – найти сумму отрицательных из них.
2. Найти в последовательности чисел количество пар соседних чисел, таких, что одно больше 100, а второе меньше –100. Окончание ввода – число 0.
3. Определить количество совпадающих цифр в одинаковых разрядах двух чисел (25589, 5019 => 2).

4. $f(x)=\begin{cases} x+\cos(x)/(u^2+2), 0<x<50 \\ \ln(2\cdot x+u), x<-2 \\ 2^x/(x+1)^4, \text{в остальных случаях} \end{cases}$

Вариант №25

1. Найти объем прямоугольного параллелепипеда по трем его сторонам. Обеспечить контроль введенных данных.
2. Найти наименьшее число среди чисел последовательности, кратных 4. Окончание ввода – количество однозначных чисел последовательности равно 5.
3. Получить новое число, уменьшая каждую четную цифру числа в 2 раза (1643702 => 1323701).

4. $f(x)=\begin{cases} \sqrt{u\cdot x^5}, 0<x<4 \\ \cos(x+u), x>30 \\ e^x/x^6, \text{в остальных случаях} \end{cases}$

ЛАБОРАТОРНАЯ РАБОТА №2

«Массивы»

Условия и ограничения для работы №2

1. Исходные данные во всех задачах типа int.
2. Обеспечить контроль корректности ввода размерности исходных массивов. При неверном вводе пользователь должен иметь возможность вводить размерность до верного ввода.
3. Если это подразумевается заданием, реализовать вывод промежуточных результатов.
4. В задаче 2.2 реализовать вывод результатов до и после сортировки.

Вариант №1

1. Дан массив A[n]. Найти среднее арифметическое элементов массива, кратных заданному числу k. Если это возможно, поменять местами первые 3 и последние 3 элементы массива. Добавить в начало массива элемент, совпадающий с последним элементом.
2. Дан массив A[n][m]. Удалить из массива A последнюю строку, в которой содержится максимальный элемент массива. Сформировать из удалённой строки одномерный массив B. Отсортировать массив B по возрастанию.

Вариант №2

1. Дан массив A[n]. Найти, сколько раз в массиве встречается минимальный элемент. Найти все индексы минимального элемента в массиве. Удалить из массива первый по порядку минимальный элемент.
2. Дан массив A[n][m]. Сформировать одномерный массив B из неотрицательных элементов строк массива A, в которых содержится минимальный элемент массива A. Отсортировать массив B по убыванию.

Вариант №3

1. Дан массив A[n]. Найти индекс такого элемента массива, который имеет максимальную сумму предыдущего и последующего элементов. Переставить элементы массива после этого элемента в обратной последовательности.
2. Дан массив A[n][n]. Добавить в массив A строку, каждый элемент которой равен количеству нулевых элементов в соответствующем столбце. Сформировать одномерный массив B из средних арифметических значений строк массива A. Отсортировать массив B по возрастанию.

Вариант №4

1. Дан массив A[n]. Преобразовать массив таким образом, чтобы сначала располагались все элементы, имевшие нечетные индексы, а затем элементы с четными индексами. Удалить из массива все нулевые элементы.
2. Дан массив A[n][n]. Определить индексы первого и последнего столбцов массива, все элементы которых одинаковые между собой, и поменять их местами. Отсортировать главную диагональ массива по возрастанию абсолютных величин.

Вариант №5

1. Дан массив A[n]. Найти индексы первого по порядку положительного и последнего по порядку отрицательного элементов массива. Удалить из массива все нечетные элементы между элементами с этими индексами.
2. Дан массив A[n][n]. Сформировать одномерный массив B из элементов главной диагонали тех строк массива A, в которых отсутствует нулевой элемент. Удалить из массива A первую строку и последний столбец. Отсортировать массив B по возрастанию.

Вариант №6

1. Дан массив A[n]. Определить, имеются ли в массиве два идущих подряд нулевых элемента.

Первую пару нулевых элементов заменить минимальным и максимальным элементами. Удалить из массива, если это возможно, по одному элементу до найденной пары и после.

2. Даны массивы A[n][m] и B[n]. Определить, является ли массив B столбцом массива A. Удалить из массива A столбцы, совпадающие с массивом B. Отсортировать заданную строку массива A по возрастанию.

Вариант №7

1. Дан массив A[n]. Найти наибольшее количество одинаковых идущих подряд элементов массива и сам этот элемент. Заменить все элементы массива, кратные найденному элементу, на последний элемент массива, умноженный на 10.
2. Дан массив A[n][m]. Среди столбцов массива, содержащих только такие элементы, которые по модулю не больше 10, найти столбец с минимальным произведением элементов. Отсортировать найденный столбец массива по возрастанию.

Вариант №8

1. Дан массив A[n]. Найти количество элементов, расположенных между первым по порядку минимальным и последним по порядку максимальным элементами массива. Добавить после последнего по порядку максимального элемента массива элемент с заданным значением t.
2. Дан массив A[n][m]. Заменить все элементы массива на противоположные по знаку, а затем поменять местами первый и последний столбец массива. Удалить из массива предпоследний столбец. Отсортировать первую строку массива по убыванию абсолютных величин.

Вариант №9

1. Дан массив A[n]. Если первый по порядку нулевой элемент находится левее последнего по порядку отрицательного элемента, то найти среднее арифметическое элементов, находящихся между ними. Удалить из массива все элементы, модуль которых больше найденного среднего арифметического.
2. Дан массив A[n][n]. Расположить на главной диагонали массива A для каждого столбца сумму элементов. Сформировать одномерный массив B из строки массива A, в которой элемент на главной диагонали имеет максимальной значение. Отсортировать массив B по убыванию.

Вариант №10

1. Дан массив A[n]. Найти количество элементов массива между первым по порядку нулевым и последним по порядку нулевым элементами массива. Поменять эти элементы попарно местами. Удалить из массива элементы до первого по порядку нулевого элемента.
2. Дан массив A[n][m]. Поэлементно вычесть заданную строку массива из тех строк массива, в которых нет элемента с заданным значением t. Отсортировать по убыванию строку массива, в которой больше всего нулевых элементов.

Вариант №11

1. Дан массив A[n]. Сдвинуть в конец массива все элементы, кратные заданному числу k, не меняя порядок следования остальных элементов. После последнего по порядку отрицательного элемента добавить в массив элемент со значением 100.
2. Дан массив A[n][m]. Поменять местами две заданные строки массива, предварительно переставив элементы в обратном порядке в той из них, сумма элементов в которой минимальна. Отсортировать первый столбец массива по возрастанию.

Вариант №12

1. Дан массив A[n]. Переместить все нулевые элементы в начало массива, не меняя порядок следования остальных элементов. Добавить в массив после последнего по порядку положительного элемента столько элементов с заданным значением k, сколько всего нулевых элементов есть в массиве.

2. Дан массив A[n][n]. Сформировать одномерный массив B из максимальных элементов таких столбцов массива A, в которых нет отрицательных элементов. Отсортировать массив B по убыванию.

Вариант №13

1. Дан массив A[n]. Преобразовать массив таким образом, чтобы элементы равные 0, располагались после отрицательных элементов, но перед положительными элементами. Удалить из массива все элементы с четными индексами, модуль которых не из заданного диапазона [a; b].
2. Дан массив A[n][n]. Сформировать одномерный массив B из элементов массива A, значения которых по модулю больше среднего арифметического элементов массива A из столбцов, в которых нет нулевых элементов. Отсортировать массив B по возрастанию.

Вариант №14

1. Дан массив A[n]. Найти сумму элементов массива без двух наибольших элементов. Добавить в начало и конец массива по одному элементу с заданным значением k.
2. Дан массив A[n][n]. Добавить в массив A столбец, элементы которого равны количеству положительных элементов в соответствующей строке массива. Сформировать одномерный массив B из положительных элементов массива A, меньших среднего арифметического элементов главной диагонали исходного массива A. Отсортировать массив B по возрастанию.

Вариант №15

1. Дан массив A[n]. Найти количество элементов массива с четными индексами, больших по абсолютной величине, чем k. Продублировать в массиве каждый отрицательный элемент.
2. Дан массив A[n][m]. В строках массива, содержащих только элементы из заданного диапазона [a; b], каждый элемент, кроме первого, заменить суммой всех предыдущих. Отсортировать заданный столбец массива по убыванию абсолютных величин.

Вариант №16

1. Дан массив A[n]. Удалить из массива все элементы, модуль которых не находится в интервале [a; b]. Освободившиеся в конце массива элементы заполнить числом k. Найти, сколько в массиве элементов, совпадающих с предыдущим и последующим элементами.
2. Дан массив A[n][n]. Найти среднее арифметическое элементов над главной диагональю и максимальный элемент на главной диагонали массива A. Сформировать одномерный массив B из столцов массива A, у которых элемент на главной диагонали совпадает с максимальным элементом главной диагонали. Отсортировать массив B по убыванию.

Вариант №17

1. Дан массив A[n]. Найти сумму элементов массива с нечетными индексами, расположенных между первым по порядку и последним по порядку положительными элементами. Удалить из массива все кратные 5 элементы, которые не равные 0.
2. Дан массив A[n][m]. Определить, есть ли в массиве ровно две строки только с элементами не из заданного диапазона [a; b]. Удалить обе эти строки из массива. Отсортировать заданную строку массива по убыванию.

Вариант №18

1. Дан массив A[n]. Переставить в обратной последовательности все элементы массива с нечетными индексами. Определить, являются ли элементы полученного массива после второго положительного элемента возрастающей последовательностью.
2. Дан массив A[n][m]. Сформировать одномерный массив B из четных элементов массива A из заданного диапазона [a; b]. Отсортировать по убыванию в массиве A все строки, в которых содержится минимальный элемент массива A.

Вариант №19

1. Дан массив A[n]. Найти индексы двух соседних элементов массива, сумма которых максимальна. Если таких пар элементов несколько, то удалить из массива все такие пары кроме последней.
2. Дан массив A[n][n]. Найти все индексы строк массива, для которых произведение нулевого и диагонального элементов максимально. Удалить из массива первую строку. Отсортировать заданный столбец массива по убыванию.

Вариант №20

1. Дан массив A[n]. Найти произведение абсолютных величин нечетных элементов массива с четными индексами. После каждого нулевого элемента массива добавить элемент с заданным значением k.
2. Дан массив A[n][n]. Найти количество столбцов массива A, в которых содержатся только положительные элементы из заданного диапазона [a; b], но не содержатся нулевые элементы. Сформировать одномерный массив B из индексов найденных столбцов. Отсортировать побочную диагональ массива A по убыванию.

Вариант №21

1. Дан массив A[n]. Поменять местами максимальный элемент среди отрицательных элементов с минимальным элементом среди положительных элементов. Удалить из массива все элементы с нечетными индексами, меньшие чем предыдущий элемент.
2. Дан массив A[n][n]. Заменить в массиве элементы побочной диагонали суммами модулей элементов соответствующих столбцов. Поменять местами первую и последнюю строки массива. Отсортировать главную диагональ массива по возрастанию.

Вариант №22

1. Дан массив A[n]. Проверить, верно ли утверждение, что произведение положительных элементов массива с индексами от k до m кратно числу p. Если утверждение верно, удалить из массива все эти элементы. Иначе добавить в начало массива p элементов, равных 0.
2. Дан массив A[n][n]. Сформировать одномерный массив B путем поэлементного умножения строки и столбца массива A, где находится его максимальный элемент. Отсортировать массив B по убыванию.

Вариант №23

1. Дан массив A[n]. Найти сумму элементов массива, таких, что индекс элемента кратен заданному числу k, а сам элемент не кратен k. Переставить попарно элементы массива до второго по порядку положительного элемента массива. Удалить из массива все начальные идущие подряд отрицательные элементы.
2. Дан массив A[n][n]. Удалить из массива столбец, в котором больше всего нулевых элементов (если таких столбцов несколько, то удалить все). Отсортировать последнюю строку массива по возрастанию.

Вариант №24

1. Дан массив A[n]. Переставить все элементы массива в обратном порядке. Найти, сколько раз в массиве встречается максимальный элемент. Добавить после каждого максимального элемента элемент с заданным значением t.
2. Дан массив A[n][n]. Определить, совпадают ли в массиве A главная и побочная диагонали. Сформировать одномерный массив B из элементов строки и столбца матрицы, где находится ее максимальный элемент. Отсортировать массив B по возрастанию.

Вариант №25

1. Дан массив A[n]. Найти общее количество элементов, расположенных до первого по порядку

элемента кратного 5 и после последнего по порядку элемента кратного 5. Переместить все четные элементы массива в начало массива, не меняя порядок следования остальных элементов.
2. Дан массив A[n][m]. Найти в массиве A столбец с минимальной суммой положительных элементов. Сформировать одномерный массив B из этого столбца. Отсортировать массив B по возрастанию.

ЛАБОРАТОРНАЯ РАБОТА №3

«Указатели»

Условия и ограничения для работы №3

1. Выполнить с помощью указателей задачу 1.1. Переменные можно описывать только для резервирования под них места в памяти, а работа с этими переменными должна выполняться с помощью указателей.
2. Выполнить задачу 2.1, используя для доступа к элементам массивов указатели (операцию [] использовать нельзя).
3. Выполнить задачу 2.2, используя для доступа к элементам массивов указатели (операции [] и [][] использовать нельзя).

ЛАБОРАТОРНАЯ РАБОТА №4

«Строки»

Условия и ограничения для работы №4

1. В задачах все исходные данные и результаты являются строками и должны вводиться и выводиться как строки, а не как отдельные символы.
2. Задачи 4.1 и 4.2 выполнить, не используя функции библиотек <string.h> и <mem.h>.
3. Задачу 4.3 выполнить, используя функции библиотеки <string.h>. Посимвольное обращение к элементам строки допустимо только для проверки на '\0' и для установки '\0'.
4. В задаче 4.3 не требовать ввода количества слов в массиве. Признаком окончания ввода массива слов является пустая строка.
5. Предложение – это строка, состоящая из слов, разделенных одним пробелом. Знаки препинания считать частью слова. Окончание предложения – символ '\0'.
6. Задание «сформировать предложение или слово (массив слов)» подразумевает, что предложение или слово (массив слов) должны быть сформировано как строки (массив строк), а не только выведены на экран.

Вариант №1

1. Дана строка. Добавить в начало строки столько последних символов строки, сколько всего есть таких символов в строке ("dadfaedsa" => "aaadadfaedsa").
2. Дано предложение. Найти слова, начинающиеся на заданную букву и состоящие из 4-х букв, сформировав из них массив слов.
3. Дан массив слов и подстрока. Сформировать предложение из слов, имеющих в своем составе после 2-го символа только символы из заданной подстроки, предварительно удалив из таких слов первые 2 символа.

Вариант №2

1. Дана строка. Вставить между одинаковыми символами строки символ '=', а между разными – символ '*' ("aadfersss" => "a=a*d*f*e*r*s=s=s").
2. Дано предложение. Найти в предложении слова, имеющие в своем составе подряд 3 одинаковые заданные буквы. Сформировать из этих слов массив слов. Преобразовать массив слов, удалив из слов все вхождения заданной буквы.
3. Дан массив слов. Преобразовать массив, вставив в каждое слово длиной более 2-х символов после 2-ой буквы подстроку из 2-х начальных букв этого же слова. Сформировать предложение из тех слов массива, которые являются перевертышами.

Вариант №3

1. Дана строка. Заменить в строке все маленькие латинские буквы от 'a' до 'y' на следующую по алфавиту, а букву 'z' удалить ("abc2=zx0" => "bcd2=y0").
2. Дано предложение. Удалить из него слова, состоящие не менее чем из 4-х букв, хотя бы одна из которых латинская буква 'w'.
3. Дан массив слов и подстрока. Сформировать предложение из слов заданной длины, в которых нет перевернутой заданной подстроки, но есть сама заданная подстрока. Перед включением в предложение удалить из слов первое вхождение заданной подстроки.

Вариант №4

1. Дана строка. Переставить все символы строки в обратном порядке и добавить в начало и конец строки символ '?' ("abcd" => "?dcba?").
2. Дано предложение. Удалить из предложения слова длиной 1 или 2 символа. В словах из 3-х символов вставить между всеми буквами символ '*'.
3. Дан массив слов и подстрока. Сформировать предложение из слов, содержащих заданную подстроку не более 2-х раз (каждая буква может входить только в одну подстроку). В предложение должны войти слова только с четной длиной.

Вариант №5

1. Дана строка. Удалить из строки все символы '*', а перед каждым символом '?' вставить подстроку "ab" ("a?*5b?!*??d" => "aab?5bab?!ab?ab?d").
2. Дано предложение. В каждом слове предложения повторяющиеся цепочки букв длиной менее 3-х букв дополнить до 3-х букв, а более длинные – усечь до 3-х букв.
3. Дан массив слов и подстрока. Сформировать два предложения из слов массива. В первое предложение должны войти слова, которые имеют в своем составе подстроку в конце слова. Во второе предложение должны войти слова, которые не имеют в своем составе подстроку с начала слова. Слова в первом предложении должны быть отсортированы по длине слова, а во втором – по алфавиту.

Вариант №6

1. Дана строка. Вставить в строке после каждой цифры символ '!', а перед всеми остальными символами – символ '?' ("a7bc5" => "?a7!?b?c5!").
2. Дано предложение. Сформировать массив из слов предложения, которые состоят из различных букв.
3. Дан массив слов. Сформировать предложение из слов, в которых k первых букв совпадают с k последними буквами, взятыми в обратном порядке.

Вариант №7

1. Дана строка. Повторяющиеся подряд символы строки удалить, оставив лишь один из них ("abbcccdee" => "abcde").
2. Дано предложение. Сформировать массив слов из таких слов предложения, которые имеют в своем составе латинскую букву 'a', обрамленную цифрами.

3. Дан массив слов, каждое из которых начинается с 3-х цифр. Сформировать предложение из слов, длина которых более 5 символов, и последние 4 символа которых не содержат подстроку "**", упорядочив в предложении слова по возрастанию первых 3-х символов.

Вариант №8

1. Дана строка. Определить, состоит ли строка из различных неповторяющихся символов ("abcd5" => "да")).
2. Дано предложение. Изменить предложение, удалив в каждом чётном слове цифры, а каждое нечётное слово, содержащее более 4-х символов, укоротив до 4-х символов.
3. Дан массив слов и слово. Сформировать предложение из слов, которые содержат в своем составе после 3-го символа перевернутое заданное слово, упорядочив слова в предложении по убыванию количества символов в словах.

Вариант №9

1. Дана строка. Удалить с конца строки символы так, чтобы длина строки стала кратна трем. Разбить строку на группы по три символа. В каждой группе символы переставить в обратном порядке ("abc123sh" => "cba321").
2. Дано предложение. Удалить из предложения слова, в которых есть повторяющиеся символы.
3. Дан массив слов и предложение. Сформировать новое предложение из слов массива, которые не входят в заданное предложение, причем во всех словах, длина которых строго больше 4, удалить по 2 символа в начале и в конце слова.

Вариант №10

1. Дана строка. Переставить части строки относительно первой цифры в строке ("abc5de7y" => "de7y5abc").
2. Дано предложение и слово. Сформировать массив слов из таких слов предложения, которые получены простой перестановкой символов заданного слова.
3. Дан массив слов. Сформировать предложение из слов, у которых в начале и конце слова есть совпадающие непересекающиеся подстроки длиной 3 символа, предварительно укоротив такие слова на 3 первых символа.

Вариант №11

1. Дана строка. Заменить в строке все подстроки "ab" на символ '*' ("ab2*dacab7" => "*2*dac*7").
2. Дано предложение. Удалить из предложения слова с четной длиной, а в словах с нечетной длиной переставить символы в обратной последовательности.
3. Дан массив слов. Сформировать предложение из слов, в составе каждого из которых дважды встречается или подстрока "****", или подстрока "++", причем встречающиеся подстроки не пересекаются.

Вариант №12

1. Дана строка. Найти в строке длину самой длинной подстроки из одинаковых символов ("aaabccccccdd" => 5).
2. Дано предложение. Вставить в предложении между всеми одинаковыми словами слово "=".
3. Дан массив слов. Сформировать предложение из слов, которые не являются перевертышами и не состоят из одних цифр.

Вариант №13

1. Дана строка. Если длина строки кратна 3, разбить строку на группы по три символа. В группе средний символ удалить, а крайние поменять местами ("abc123shr" => "ca31rs").
2. Дано предложение. Найти в предложении слова, первый и последний символы в которых одинаковы. Сформировать из них массив слов, удалив в каждом таком выделенном слове первый и последний символы.

3. Дан массив слов. Преобразовать массив слов, укоротив слова с начала слова на количество символов в предыдущем слове, если длина предыдущего слова меньше. Сформировать предложение из слов массива, длина которых четная.

Вариант №14

1. Дана строка. Удалить из строки все символы '?', добавив в конец строки символ '!' ("?ab?bc?" => "abbc!").
2. Дано предложение. Сформировать массив из слов, в составе которых есть группы повторяющихся символов длиной более 2-х символов.
3. Дан массив слов. Сформировать предложение из слов, длина которых больше длины предыдущего слова, но меньше длины последующего, сместив их символы по кругу на 3 позиции вправо.

Вариант №15

1. Дана строка. Если в строке ровно два символа '!', удалить в строке все символы между символами '!' ("a!5b?!d" => "a!!d").
2. Дано предложение. Записать все его слова через черточку, удалив из предложения слова из 3-х символов.
3. Дан массив слов и подстрока. Преобразовать массив слов, удалив из слов все группы цифр длиной больше 2-х. Сформировать предложение из слов, которые заканчиваются заданной подстрокой.

Вариант №16

1. Дана строка. Найти в строке индексы первого и последнего символов, таких, что слева и справа от них стоят цифры ("a7b8cd5ef7g1" => 2 и 10).
2. Дано предложение. Сформировать новое слово из первых и последних символов таких слов предложения, которые не содержат цифр. Добавить полученное слово в предложение после любого самого длинного слова.
3. Дан массив слов. Сформировать предложение из слов, длина которых более 7 символов, оставив в предложении от каждого такого слова 6 начальных и 2 конечных символа, поставив между ними символ '+'.

Вариант №17

1. Дана строка. Переставить в строке символы в обратном порядке до первой цифры и после последней цифры ("abc5zx123knmt" => "cba5zx123tmnk").
2. Дано предложение. Заменить каждое слово предложения на новое, составив новое слово так, чтобы каждая буква в слове повторялась один раз, сохранив общий порядок следования букв.
3. Дан массив слов. Сформировать предложение, в котором сначала идут односимвольные, затем двухсимвольные и т.д. слова, причем в предложении слова одинаковой длины должны быть отсортированы по алфавиту.

Вариант №18

1. Дана строка. Заменить в строке все цифры, кроме цифры 0, на подстроку "++", а цифру 0 заменить на символ '*' ("7a5b021d" => "++a++b*++++d ").
2. Дано предложение. Поменять в предложении местами первое слово, начинающееся с цифры, и последнее слово, заканчивающееся на цифру.
3. Дан массив слов. Сформировать предложение из слов, длина которых больше 2-х символов, циклически сместив в словах символы на 2 позиции влево. Слова в предложении должны быть отсортированы по алфавиту.

Вариант №19

1. Дана строка. Удалить из строки все цифры, а символы '+' и '*' удвоить ("a+5b*c21++d" =>

"a++b**c++++d").
2. Дано предложение. Удалить во всех словах предложения каждый третий символ, сформировав из всех удаленных символов новое слово.
3. Дан массив слов и слово. Сформировать предложение из таких слов массива, которые имеют в своем составе только символы из заданного слова и длина которых больше 2-х символов. Слова в предложении должны быть отсортированы по первым 3-м символам по алфавиту.

Вариант №20

1. Дана строка. Если строка не заканчивается на символ '!', вставить в строке перед каждой цифрой символ '?' ("a7bc5" => "a?7bc?5").
2. Дано предложение. Найти в предложении слова, которые содержат в своем составе самое короткое слово этого предложения (если есть несколько одинаковых по длине самых коротких слов, то брать первое из них по порядку в предложении). Сформировать из найденных слов массив слов.
3. Дан массив слов и подстрока. Сформировать новое слово из 2-х начальных букв тех слов, где подстрока повторяется по меньшей мере 2 раза после третьей буквы.

Вариант №21

1. Дана строка. Если первый символ строки совпадает с последним символом, удалить из строки, если это возможно, два первых и два последних символа ("atbcda" => "bc").
2. Дано предложение. Сократить в предложении каждое слово с длиной более 4-х букв, поставив после 4-й буквы точку, а остаток отбросив.
3. Дан массив слов и подстрока. Каждое слово массива с четным индексом преобразовать, удвоив его. Сформировать предложение из слов, которые содержат заданную подстроку после k-го символа.

Вариант №22

1. Дана строка. Если строка не заканчивается на символ '!', перед и после каждого символа '?' добавить символ '_' ("a?5b??d" => "a_?_5b_?__?_d").
2. Дано предложение и слово из 3-х символов. Найти слова предложения, содержащие заданное слово, сформировав из них массив слов.
3. Дан массив слов. Сформировать предложения из слов, в которых первые k букв совпадают. В результате – массив предложений.

Вариант №23

1. Дана строка. Если первый символ строки цифра, вставить ее после каждого знака '+' в строке ("7b+cdf+e+" => "7b+7cdf+7e+7").
2. Дано предложение. Сформировать массив из слов предложения, в составе которых есть цифры, предварительно удалив из слов эти цифры.
3. Дан массив слов. Сформировать предложение из тех слов массива, в которых k последних букв не равны k первым буквам следующего слова, взятым в обратном порядке.

Вариант №24

1. Дана строка. Удалить каждый третий символ строки, добавив между всеми оставшимися символами символ '*' ("abcdefg" => "a*b*d*e*g").
2. Дано предложение. Сформировать массив слов из таких слов предложения, которые состоят только из цифр.
3. Дан массив слов и слово. Сформировать предложение из слов, входящих в заданное слово не более 1-го раза (одна буква может входить только в одно слово).

Вариант №25

1. Дана строка. Найти общее количество символов 'a' и 'b' в строке. Если это количество четное,

удалить из строки символ 'a', в противном случае – удалить символ 'b' ("abbcdafb" => "acdaf").
2. Дано предложение. Удалить в предложении в словах длиной более 4-х символов 2 первых символа, добавив между всеми оставшимися в слове символами символ '*'.
3. Дан массив слов. Получить строку, взяв от каждого слова длиной более 3-х символов по 2 конечных символа. Сформировать предложение из тех слов массива, которые содержат только символы из образованной строки.

ЛАБОРАТОРНАЯ РАБОТА №5

«Функции»

5.1. Написать две функции для вычисления заданной величины и их вызов из функции main().

Условия и ограничения для работы №5.1

1. Исходные данные должны вводиться в функции main().
2. Первая функция должна возвращать заданную величину.
3. Во второй функции обеспечить контроль правильности исходных данных.
4. Вторая функция, кроме вычисления заданной величины, должна возвращать признак правильности исходных данных.
5. В функции main() вывести результат работы первой функции и результат работы второй функции или сообщение об ошибке.
6. Глобальные переменные в программе использовать нельзя.

Вариант №1

Длина гипотенузы прямоугольного треугольника по двум катетам ($c = \sqrt{a^2 + b^2}$).

Вариант №2

Координата середины отрезка [x_1; x_2] на прямой ($x_{cp} = \frac{x_1 + x_2}{2}$).

Вариант №3

Сумма n первых членов арифметической прогрессии ($S_n = \frac{(a_1 + a_n) \cdot n}{2}$).

Вариант №4

Площадь ромба по стороне и высоте ($S = ah$).

Вариант №5

Площадь трапеции по основаниям и высоте ($S = \frac{a+b}{2} h$).

Вариант №6

Расстояние между двумя несовпадающими точками с положительными координатами на прямой ($r = |x_2 - x_1|$).

Вариант №7

Площадь полной поверхности цилиндра по радиусу и высоте ($S = 2\pi RH + 2\pi R^2$).

Вариант №8

Объем шара по радиусу ($V = \frac{4}{3}\pi R^3$).

Вариант №9

Площадь поверхности прямоугольного параллелепипеда по трем сторонам ($S = 2(ab + bc + ac)$).

Вариант №10

Площадь боковой поверхности цилиндра по радиусу и высоте ($S_{бок} = 2\pi RH$).

Вариант №11

Объем прямоугольного параллелепипеда по трем сторонам ($V = abc$).

Вариант №12

Сторона правильного n-угольника по радиусу описанной окружности ($a_n = 2R\sin\frac{\pi}{n}$).

Вариант №13

Площадь круга по диаметру ($S = \frac{\pi d^2}{4}$).

Вариант №14

Площадь ромба по его диагоналям ($S = \frac{1}{2}d_1 d_2$).

Вариант №15

Объем пирамиды по площади основания и высоте ($V = \frac{1}{3}S_{осн}H$).

Вариант №16

Сумма внутренних углов выпуклого n-угольника ($S = \pi(n-2)$).

Вариант №17

Доля в процентах одного положительного числа от второго ($p = 100\frac{x}{y}$).

Вариант №18

Объем конуса по высоте и радиусу основания ($V = \frac{1}{3}\pi R^2 H$).

Вариант №19

Длина диагонали прямоугольного параллелепипеда по сторонам ($d = \sqrt{a^2 + b^2 + c^2}$).

Вариант №20

Расстояние между двумя несовпадающими точками на плоскости ($r = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$).

Вариант №21

Периметр прямоугольника по сторонам ($P = 2(a + b)$).

Вариант №22

Значение n-го члена арифметической прогрессии с положительным первым членом ($a_n = a_1 + d(n-1)$).

Вариант №23

Длина окружности по радиусу ($C = 2\pi R$).

Вариант №24

Длина второй диагонали ромба по стороне и первой диагонали ($d_2 = \sqrt{4a^2 - {d_1}^2}$).

Вариант №25

Длина средней линии трапеции по основаниям ($c = \frac{a+b}{2}$).

5.2. Выполнить задачу 5.1 в виде двух единиц компиляции: один исходный файл с функцией main(), второй – с разработанными функциями.

Условия и ограничения для работы №5.2

1. Исходные данные необходимо получать из командной строки (параметры функции main()).
2. В программе должен выполняться контроль количества задаваемых параметров командной строки.

5.3. Написать функцию для обработки одномерного массива для задачи 2.1 и ее вызов из функции main().

Условия и ограничения для работы №5.3

1. Ввод размерности массива и самого массива должен осуществляться из текстового файла в функции main().
2. Результаты работы программы (имя исходного файла, входные и выходные данные теста) в функции main() вывести в текстовый файл произвольного формата.
3. Имена исходного и результирующего файлов задавать в командной строке.

5.4. Написать функцию для обработки символьной информации для задачи 4.1 и ее вызов из функции main().

Условия и ограничения для работы №5.4

1. Строки для обработки передаются в программу как параметры командной строки.
2. Если это необходимо по условию задачи, все остальные исходные данные должны вводиться с клавиатуры.
3. Разработанная функция должна вызываться столько раз, сколько параметров передано в функцию main().

ЛАБОРАТОРНАЯ РАБОТА №6

«Динамическое выделение памяти»

6.1. Для заданного объекта описать шаблон структуры в соответствии с заданием

варианта и выполнить следующие действия:

1) описать структурную переменную, заполнить ее с клавиатуры и вывести ее на экран;
2) динамически создать структурную переменную, заполнить ее с клавиатуры и вывести ее на экран, удалить созданную переменную;
3) описать одномерный массив структурных переменных, заполнить его с клавиатуры, выполнить требуемые действия в соответствии с заданием варианта (А), вывести на экран полученные результаты. Для доступа к элементам массива использовать указатели (операцию [] использовать нельзя);
4) ввести размерность массива, динамически создать одномерный массив структурных переменных, заполнить его с клавиатуры, выполнить требуемые действия в соответствии с заданием варианта (Б), вывести на экран полученные результаты, удалить созданный массив.

Для ввода с клавиатуры и вывода на экран отдельной структурной переменной должны быть разработаны функции. При вводе структурной переменной необходимо выдавать подсказку для типа объекта, а при выводе – выполнять расшифровку типа объекта.

Вариант №1

Объект – книга, множество данных – книги в библиотеке:

1) название (char[ ]);
2) год издания (int);
3) тип: детская, научная, художественная, техническая, другое (int).

А – отсортировать книги по возрастанию названия, книги с одинаковым названием должны быть отсортированы по убыванию года издания;

Б – найти количество книг заданного типа, год издания которых равен максимальному году издания среди всех книг.

Вариант №2

Объект – школа, множество данных – школы города:

1) номер школы (char[ ]);
2) количество учащихся (long);
3) тип: лицей, общеобразовательная, гимназия, другое (int).

А – отсортировать школы по убыванию количества учащихся, школы с одинаковым количеством учащихся должны быть отсортированы по типу;

Б – найти общее количество лицеев и гимназий с количеством учащихся из заданного диапазона.

Вариант №3

Объект – наблюдение за климатом, множество данных – наблюдения за климатом в городе:

1) дата (char[ ]);
2) температура (int);
3) вид осадков: нет, дождь, снег, град (int).

А – отсортировать наблюдения по виду осадков, наблюдения с одинаковым видом осадков должны быть отсортированы по убыванию температуры;

Б – найти среднюю температуру дождливых дней заданного года.

Вариант №4

Объект – городской транспорт, множество данных – общественный транспорт города:

1) номер маршрута (char[ ]);
2) длина маршрута в километрах (float);

3) вид: маршрутное такси, автобус, троллейбус, трамвай (int).

А – отсортировать транспорт по возрастанию длины маршрута, транспорт с одинаковой длиной маршрута должен быть отсортирован по номеру маршрута;

Б – найти маршрут заданного вида, длина которого максимально отличается от средней длины автобусных маршрутов.

Вариант №5

Объект – газета, множество данных – газеты города:

1) название (char[ ]);
2) год основания (int);
3) периодичность выхода: ежедневная, 5 раз в неделю, 1 раз в неделю, 1 раз в месяц (int).

А – отсортировать газеты по убыванию года основания, газеты с одинаковым годом основания должны быть отсортированы по убыванию названия;

Б – найти минимальную разность в годах основания двух еженедельных газет.

Вариант №6

Объект – самолет, множество данных – самолеты в аэропорту:

1) название марки (char[ ]);
2) количество мест (int);
3) примечание: принадлежит аэропорту, взят в аренду, личный, другое (int).

А – отсортировать самолеты по возрастанию названия марки, самолеты с одинаковым названием марки должны быть отсортированы по возрастанию количества мест;

Б – для взятых в аренду самолетов найти общее количество самолетов заданной марки.

Вариант №7

Объект – изучаемый предмет, множество данных – изучаемые предметы на факультете:

1) количество часов (int);
2) название (char[ ]);
3) отчетность: зачет, дифференцированный зачет, экзамен, реферат, нет (int).

А – отсортировать изучаемые предметы по виду отчетности, предметы с одинаковым видом отчетности должны быть отсортированы по убыванию названия;

Б – среди предметов, по которым нет отчетности, и название которых содержит заданную букву, найти предмет с минимальным количеством часов.

Вариант №8

Объект – дом, множество данных – дома населенного пункта:

1) название улицы (char[ ]);
2) номер дома (int);
3) тип: многоквартирный, частный, арендный, другое (int).

А – отсортировать дома по убыванию названия улицы, дома одной улицы должны быть отсортированы по возрастанию номера;

Б – найти количество различных типов домов, номер которых находится в заданном диапазоне.

Вариант №9

Объект – абитуриент, множество данных – список абитуриентов вуза:

1) фамилия абитуриента (char[ ]);
2) средний балл аттестата (float);
3) категория зачисления: общий конкурс, целевой прием, вне конкурса, без экзаменов (int).

А – отсортировать абитуриентов по категории зачисления, абитуриенты одинаковой категории должны быть отсортированы по убыванию среднего балла аттестата;

Б – для каждой категории зачисления найти среднюю длину фамилии абитуриентов.

Вариант №10

Объект – собака, множество данных – собаки на выставке:

1) возраст (int);
2) кличка (char[ ]);
3) вид: служебная, декоративная, охотничья, другая (int).

А – отсортировать собак по кличкам, собаки с одинаковыми кличками должны быть отсортированы по возрастанию возраста;

Б – найти количество различных кличек собак заданного возраста.

Вариант №11

Объект – вуз, множество данных – вузы страны:

1) год создания (int);
2) название (char[ ]);
3) тип: университет, академия, институт (int).

А – отсортировать вузы по убыванию года создания, вузы с одинаковым годом создания должны быть отсортированы по типу;

Б – найти самый старый институт, название которого заканчивается на заданную букву.

Вариант №12

Объект – улица, множество данных – улицы города:

1) длина в километрах (float);
2) название (char[ ]);
3) тип: проспект, улица, пешеходная, переулок, проезд (int).

А – отсортировать улицы по убыванию длины, улицы одной длины должны быть отсортированы по названию;

Б – найти общую длину переулков и пешеходных улиц, в названии которых есть цифры.

Вариант №13

Объект – стадион, множество данных – стадионы страны:

1) название (char[ ]);
2) вместимость человек (long);
3) тип: международный, республиканский, городской, другое (int).

А – отсортировать стадионы по типу, стадионы одного типа должны быть отсортированы по возрастанию названия;

Б – для каждого типа стадиона найти среднюю вместимость.

Вариант №14

Объект – населенный пункт, множество данных – населенные пункты страны:

1) количество жителей (long);
2) название (char[ ]);
3) тип: столица, город, поселок, агрогородок, деревня, другое (int).

А – отсортировать населенные пункты по типу, населенные пункты с одинаковым типом должны быть отсортированы по названию;

Б – найти количество населенных пунктов заданного типа, количество жителей в которых меньше среднего количества жителей во всех городах.

Вариант №15

Объект – телепередача, множество данных – популярные телепередачи:

1) название (char[ ]);
2) рейтинг (int);
3) жанр: политика, экономика, кино, музыка, юмор, спорт, другое (int).

А – отсортировать телепередачи по возрастанию рейтинга, телепередачи с одинаковым рейтингом

должны быть отсортированы по названию;
Б – найти максимальный и минимальный рейтинг спортивных передач.

Вариант №16

Объект – спортсмен футбольной команды, множество данных – спортсмены футбольной команды:

1) фамилия (char[ ]);
2) рост (int);
3) амплуа: вратарь, защитник, полузащитник, нападающий (int).

А – отсортировать спортсменов по фамилии, спортсмены с одинаковой фамилией должны быть отсортированы по росту;
Б – найти средний рост спортсменов каждого амплуа.

Вариант №17

Объект – абонент мобильной связи, множество данных – абоненты оператора мобильной связи:

1) фамилия (char[ ]);
2) оплата в месяц (float);
3) подключенные услуги: интернет, роуминг, телевидение, отсутствуют (int).

А – отсортировать абонентов по фамилии, абоненты с одинаковой фамилией должны быть отсортированы по убыванию оплаты;
Б – найти минимальную оплату абонентов, имеющих подключенные услуги, и фамилия которых начинается с заданной буквы.

Вариант №18

Объект – группа в детском саду, множество данных – группы детского сада:

1) шифр группы (char[ ]);
2) количество детей в группе (int);
3) тип: ясельная, младшая, средняя, старшая (int).

А – отсортировать группы по убыванию количества детей в группе, группы с одинаковым количеством детей должны быть отсортированы по возрастанию шифра;
Б – найти среднее количество детей в младших и старших группах, в шифре которых есть заданный символ на первой или последней позиции.

Вариант №19

Объект – товар, множество данных – товары в продуктовом магазине:

1) название (char[ ]);
2) цена (float);
3) товарная группа: мясные, молочные, кондитерские, консервы, напитки, другое (int).

А – отсортировать товары по возрастанию цены, товары с одинаковой ценой должны быть отсортированы по товарной группе;
Б – найти товар заданной товарной группы, название которого содержит заданную букву и цена которого минимальна.

Вариант №20

Объект – плодовое дерево, множество данных – плодовые деревья в саду:

1) сорт (char[ ]);
2) высота (int);
3) месяц сбора урожая: июль, август, сентябрь, октябрь (int).

А – отсортировать деревья по возрастанию высоты, деревья одинаковой высоты должны быть отсортированы по сорту;
Б – для каждого месяца сбора урожая найти максимальную высоту деревьев.

Вариант №21

Объект – стипендия студента, множество данных – ведомость на выдачу стипендии в вузе:

1) фамилия студента (char[ ]);
2) сумма стипендии (float);
3) вид: учебная, повышенная, именная, социальная (int).

А – отсортировать ведомость по фамилии студента, студенты с одинаковой фамилией должны быть отсортированы по убыванию суммы стипендии;

Б – найти общую стипендию студентов, фамилии которых начинаются с заданной буквы и заканчиваются на заданную букву.

Вариант №22

Объект – результат участника олимпиады, множество данных – результаты городской олимпиады по информатике:

1) фамилия участника (char[ ]);
2) занятое место (int);
3) учебное заведение: лицей, гимназия, средняя школа (int).

А – отсортировать результаты олимпиады по занятому месту, участники, занявшие одинаковые места, должны быть отсортированы по фамилии;

Б – найти минимальное и максимальное место участников из учебных заведений заданного типа.

Вариант №23

Объект – банк, множество данных – банки города:

1) улица, на которой находится банк (char[ ]);
2) сумма вкладов (float);
3) тип: государственный, акционерный, частный (int).

А – отсортировать банки по типу, банки одного типа должны быть отсортированы по убыванию улицы;

Б – найти для каждого типа банка среднюю сумму вкладов для банков, расположенных на заданной улице.

Вариант №24

Объект – памятник, множество данных – памятники города:

1) название (char[ ]);
2) годовая стоимость расходов на содержание (long);
3) вид: квартал, площадь, здание, скульптура, другое (int).

А – отсортировать памятники по виду, памятники одного вида должны быть отсортированы по названию;

Б – для каждого вида памятников найти среднюю стоимость расходов на содержание.

Вариант №25

Объект – палата в больнице, множество данных – палаты в больнице города:

1) фамилия врача (char[ ]);
2) количество мест (int);
3) тип: мужская, женская, детская (int).

А – отсортировать палаты по типам, палаты одного типа должны быть отсортированы по фамилии врача;

Б – найти общее количество мест в детских палатах у врачей, чья фамилия начинается с заданной буквы.

6.2. Выполнить задачу 2.2, размещая все массивы в программе только динамически.

Условия и ограничения для работы №6.2

1. Ввод размерности массива и элементов массива осуществлять в функции main() из текстового файла, имя которого задаётся в командной строке.
2. Для сортировки массива должна быть разработана функция.
3. Для тестирования подготовить не менее двух входных файлов.

ЛАБОРАТОРНАЯ РАБОТА №7

«Динамические списки»

Условия и ограничения для работы №7

1. Для заданного объекта, используя оператор typedef, описать собственный тип данных в соответствии с заданием варианта.
2. Построить однонаправленный динамический список для заданного множества данных.
3. Элементы добавлять в конец списка и вводить из текстового файла с именем, задаваемом в командной строке, и расширением .in следующей структуры:

```
поле 1;поле 2;поле 3;поле 4     // элемента списка 1
поле 1;поле 2;поле 3;поле 4     // элемента списка 2
        ...
поле 1;поле 2;поле 3;поле 4     // элемента списка n
```

4. В результате работы программы сформировать текстовый файл с именем входного файла и расширением .out с данными списка, удовлетворяющими заданному в варианте условию отбора, представленными в виде таблицы (один элемент списка – на одной строке таблицы):

Поле 1	Поле 2	Поле 3	Поле 4
Значение 11 Значение 21 …	Значение 12 Значение 22 …	Значение 13 Значение 23 …	Значение 14 Значение 24 …

5. Поле 4 должно задаваться в исходном файле и отображаться в результирующем файле в расшифрованном виде.
6. Для тестирования подготовить не менее двух входных файлов заданного формата с правдоподобной информацией.
7. В программе не должно быть переменной с количеством элементов в списке.
8. В условии отбора имеется ссылка на некоторое значение. Его задавать в командной строке.

Вариант №1

Объект – самолет, множество данных – самолеты в аэропорту:

1) максимальная высота полета (long);
2) марка (char[ ]);
3) количество мест (int);
4) примечание: принадлежит аэропорту, взят в аренду, личный, другое (char).

Условие отбора: самолеты, количество мест в которых, меньше заданного.

Вариант №2

Объект – изучаемый предмет, множество данных – изучаемые предметы на факультете:

1) номер курса (int);
2) количество часов (int);
3) название (char[ ]);
4) отчетность: зачет, дифференцированный зачет, экзамен, реферат, нет (char).

Условие отбора: предметы с заданным типом отчетности.

Вариант №3

Объект – дом, множество данных – дома населенного пункта:

1) улица (char[ ]);
2) номер дома (int);
3) количество жильцов (int);
4) тип: многоквартирный, частный, арендный, другое (char).

Условие отбора: дома заданного типа, расположенные на улицах, заканчивающихся на заданную букву.

Вариант №4

Объект – банк, множество данных – банки города:

1) адрес (char[ ]);
2) количество вкладчиков (long);
3) сумма вкладов (float);
4) тип: государственный, акционерный, частный (char).

Условие отбора: частные банки, сумма вкладов в которых больше заданной величины.

Вариант №5

Объект – собака, множество данных – собаки на выставке:

1) рост в холке (int);
2) возраст (int);
3) кличка (char[ ]);
4) вид: служебная, декоративная, охотничья, другая (char).

Условие отбора: собаки заданного вида, кличка которых начинается на заданную букву.

Вариант №6

Объект – вуз, множество данных – вузы страны:

1) год создания (int);
2) название (char[ ]);
3) количество студентов (long);
4) тип: университет, академия, институт (char).

Условие отбора: университеты, созданные в заданном году.

Вариант №7

Объект – улица, множество данных – улицы города:

1) длина (float);
2) название (char[ ]);
3) количество зданий (int);
4) тип: проспект, улица, пешеходная, переулок, проезд (char).

Условие отбора: пешеходные улицы, длина которых меньше заданной.

Вариант №8

Объект – стадион, множество данных – стадионы страны:

1) название (char[ ]);
2) вместимость человек (long);
3) основной вид спорта (char[ ]);

4) тип: международный, республиканский, городской, другое (char).

Условие отбора: городские стадионы с вместимостью, больше заданной.

Вариант №9

Объект – населенный пункт, множество данных – населенные пункты страны:

1) количество жителей (long);
2) название (char[ ]);
3) главная достопримечательность (char[ ]);
4) тип: столица, город, агрогородок, деревня (char).

Условие отбора: населенные пункты, название которых начинается с заданной буквы.

Вариант №10

Объект – телепередача, множество данных – популярные телепередачи:

1) название (char[ ]);
2) длительность в минутах (int);
3) рейтинг (float);
4) жанр: политика, экономика, кино, музыка, юмор, спорт, другое (char).

Условие отбора: телепередачи, рейтинг которых больше заданного, и в названии которых есть заданная буква.

Вариант №11

Объект – школа, множество данных – школы города:

1) номер школы (int);
2) фамилия директора (char[ ]);
3) количество учащихся (long);
4) тип: лицей, общеобразовательная, гимназия, другое (char).

Условие отбора: лицеи с заданным количеством учащихся.

Вариант №12

Объект – абонент мобильной связи, множество данных – абоненты оператора мобильной связи:

1) фамилия (char[ ]);
2) оплата в месяц (float);
3) номер телефона (long);
4) подключенные услуги: интернет, роуминг, телевидение, отсутствуют (char).

Условие отбора: абоненты с оплатой, меньше заданной.

Вариант №13

Объект – рекорд в беге на 100 метров, множество данных – мировые рекорды:

1) результат (float);
2) автор (char[ ]);
3) год установления (int);
4) пол: мужской, женский (char).

Условие отбора: мужские рекорды, результат которых входит в заданный интервал.

Вариант №14

Объект – книга, множество данных – книги в библиотеке:

1) название (char[ ]);
2) стоимость (float);
3) количество (int);
4) тип: детская, научная, художественная, техническая, другое (char).

Условие отбора: научные книги, название которых начинается на заданную букву.

Вариант №15

Объект – плодовое дерево, множество данных – плодовые деревья в саду:

1) сорт (char[ ]);
2) высота (float);
3) урожайность в килограммах (int);
4) месяц сбора урожая: июль, август, сентябрь, октябрь (char).

Условие отбора: деревья, урожай с которых собирают в заданный месяц.

Вариант №16

Объект – стипендия студента, множество данных – ведомость на выдачу стипендии в вузе:

1) фамилия студента (char[ ]);
2) сумма стипендии (float);
3) средний балл студента (float);
4) вид: учебная, повышенная, именная, социальная (char).

Условие отбора: учебные стипендии студентов, фамилия которых содержит заданную букву.

Вариант №17

Объект – результат участника олимпиады, множество данных – результаты городской олимпиады по информатике:

1) фамилия участника (char[ ]);
2) занятое место (int);
3) количество набранных баллов (int);
4) учебное заведение: лицей, гимназия, средняя школа (char).

Условие отбора: участники из лицеев, занявшие места выше заданного.

Вариант №18

Объект – абитуриент, множество данных – список абитуриентов вуза:

1) фамилия абитуриента (char[ ]);
2) год окончания школы (int);
3) средний балл аттестата (float);
4) категория зачисления: общий конкурс, целевой прием, вне конкурса, без экзаменов (char).

Условие отбора: абитуриенты вне конкурса, средний балл аттестата которых больше заданного.

Вариант №19

Объект – маршрут городского транспорта, множество данных – общественный транспорт города:

1) номер маршрута (char[ ]);
2) количество единиц транспорта на маршруте (int);
3) длина маршрута (float);
4) вид: маршрутное такси, автобус, троллейбус, трамвай (char).

Условие отбора: маршруты заданного вида, длина которых больше заданной.

Вариант №20

Объект – товар, множество данных – товары в продуктовом магазине:

1) название (char[ ]);
2) цена (float);
3) количество единиц (int);
4) товарная группа: мясные, молочные, кондитерские, консервы, напитки, другое (char).

Условие отбора: мясные товары, цены которых больше заданной.

Вариант №21

Объект – наблюдение за климатом, множество данных – наблюдения за климатом в городе:

1) дата (char[ ]);
2) атмосферное давление (long);
3) температура (float);
4) вид осадков: нет, дождь, снег, град (char).

Условие отбора: дождливые дни, температура в которые была выше заданной.

Вариант №22

Объект – предприятие, множество данных – предприятия города:

1) количество сотрудников (int);
2) название (char[ ]);
3) занимаемая площадь (float);
4) вид собственности: частное, акционерное, государственное (char).

Условие отбора: частные предприятия, количество сотрудников на которых меньше заданного.

Вариант №23

Объект – группа в детском саду, множество данных – группы детского сада:

1) шифр группы (char[ ]);
2) количество детей в группе (int);
3) фамилия воспитателя (char[ ]);
4) тип: ясельная, младшая, средняя, старшая (char).

Условие отбора: ясельные группы, в которых количество детей больше заданного.

Вариант №24

Объект – музыкальный альбом, множество данных – музыкальные альбомы исполнителя:

1) название альбома (char[ ]);
2) длительность (float);
3) количество записей (int);
4) тип музыки: классическая, для детей, джаз, рок, популярная (char).

Условие отбора: альбомы, длительность которых меньше средней длительности альбомов с классической музыкой.

Вариант №25

Объект – газета, множество данных – газеты города:

1) название (char[ ]);
2) тираж (long);
3) год основания (int);
4) периодичность выхода: ежедневная, 5 раз в неделю, 1 раз в неделю, 1 раз в месяц (char).

Условие отбора: газеты, основанные в заданный период.

ЛАБОРАТОРНАЯ РАБОТА №8

«Рекурсивная обработка каталогов»

Условия и ограничения для работы №8

1. Реализовать решение лабораторной работы, используя рекурсивную функцию.
2. По окончании работы программа должна возвратиться в тот каталог, откуда она стартовала (текущий каталог).
3. Чтобы не испортить свой каталог, в компьютерном классе отлаживать задачу на диске D.
4. Ограничить поиск текущим каталогом и его внутренними подкаталогами всех уровней при отсутствии других указаний.

5. Задание вида «Удалить файлы в каталоге TMP» подразумевает удаление файлов только в каталоге TMP. В подкаталогах каталога TMP файлы удалять не надо, если в задании не указано, что и подкаталоги надо обрабатывать.
6. При возникновении конфликтных ситуаций выдавать сообщение об ошибке, операцию не производить, после чего продолжать выполнение программы.
7. Если в задании надо что-то удалить, рекомендуется в процессе отладки сначала выводить на экран то, что подлежит удалению, а не удалять.

Вариант №1

Найти самый большой по объему каталог без учета вложенных подкаталогов.

Вариант №2

Удалить из каталога файлы, расширение которых содержит менее 3-х символов, если количество файлов в каталоге больше 5.

Вариант №3

Если в каталоге существуют одновременно на одном уровне каталог и файл с одинаковыми именами, не учитывая расширения, то переместить файл в этот каталог.

Вариант №4

Переместить в текущий каталог файлы, чье расширение совпадает с именем текущего каталога.

Вариант №5

Переместить все файлы, меньшие 1К, в каталог TMP текущего каталога.

Вариант №6

Скопировать содержимое всех найденных каталогов с именем TMP в корневой каталог, переименовав их TMP1, TMP2, ...

Вариант №7

Оставить в любом каталоге не более 5 файлов, а остальные удалить. В первую очередь удалять файлы с расширением .bak.

Вариант №8

Найти все каталоги, в имени которых есть строка "TMP", созданные после 14:30 не позднее, чем за 10 дней от текущей даты, и вывести в файл list.txt пути к ним.

Вариант №9

Удалить все каталоги с именем MY_TEMP, создав вместо них пустые каталоги EMPTY.

Вариант №10

Найти каталог с заданным пользователем именем и создать в нем k вложенных друг в друга подкаталогов MY1, ..., MYk.

Вариант №11

Переместить все файлы с расширением .bak, размер которых меньше k байт, из всех подкаталогов текущего каталога в подкаталог BAK текущего каталога.

Вариант №12

Найти такие каталоги, все файлы в которых имеют время последнего обновления от 8:00 до 16:00.

Вариант №13

Найти подкаталог, в котором находится файл наибольшего размера.

Вариант №14

Найти каталог с наибольшим количеством подкаталогов.

Вариант №15

Начиная с текущего каталога, удалить из всех каталогов файлы с расширением .bak и вывести на экран самый длинный по названию путь к удаленным файлам.

Вариант №16

Найти на текущем диске заданный каталог и создать в нем файл list.txt с полными именами всех файлов, которые имеются в этом каталоге и всех его подкаталогах.

Вариант №17

Найти каталоги, в названии которых содержится заданная подстрока и суммарный объем файлов в которых превышает 1К.

Вариант №18

Если в каталоге имя некоторого файла, не учитывая расширения, совпадает с именем самого каталога, то переместить каталог в корень рабочего диска.

Вариант №19

Найти каталог, содержащий наибольшее количество файлов. Если их несколько, то найти тот, который создан ранее.

Вариант №20

Удалить все файлы, большие 50 байт, находящиеся в каталогах, начинающихся с последовательности символов TMP.

Вариант №21

Удалить все подкаталоги, имя которых начинается с MY. Для каждого удаляемого подкаталога вывести на экран полный путь к нему и количество удаленных из него файлов.

Вариант №22

Найти самый длинный по вложенности путь от текущего каталога и продублировать его структуру от корневого каталога диска D.

Вариант №23

Удалить все файлы с заданным расширением, находящиеся в каталогах, начинающихся с цифры.

Вариант №24

Удалить все файлы, созданные до 12:00, имя которых содержит цифру.

Вариант №25

Удалить во всех подкаталогах текущего каталога файлы с расширением .txt, последняя дата модификации которых меньше заданной даты дд.мм.гггг.

ЛАБОРАТОРНАЯ РАБОТА №9

«Логические операции и биты»

Условия и ограничения для работы №9

1. Входная информация – строка символов, введенная с клавиатуры.
2. Строку следует рассматривать, как единую последовательность бит ('\0' в строку не входит).
3. Нельзя преобразовывать биты в строку из 0 и 1 или числовой массив из 0 и 1.
4. При необходимости последний байт полученной строки дополнить битами 0.
5. Нумерация байт – с 0 слева направо (соответствует индексу байта в строке), нумерация бит – с 0 справа налево (младший бит имеет номер 0, старший – номер 7).
6. Результаты работы программы вывести в файл в формате: исходная строка в шестнадцатеричном формате, полученная строка в шестнадцатеричном формате и полученный результат.

Вариант №1

Если в первом байте строки количество битов 0 равно количеству битов 1, то инвертировать биты в младшем полубайте четных байтов и в старшем полубайте нечетных байтов строки (байты нумеруются с 0 – четный байт). Сложить по модулю 2 все биты не инвертированных полубайтов.

Вариант №2

Удалить из каждого байта строки первый (старший) бит при условии, что в этом байте количество битов 1 больше чем битов 0.

Вариант №3

Удалить все биты 0 в каждом байте строки, начинающемся и заканчивающемся на разные биты.

Вариант №4

Удалить в нечётных байтах (байты нумеруются с 0 – четный байт) идущие подряд биты 0, оставив лишь один из них, а в чётных байтах – идущие подряд биты 1, оставив лишь один из них.

Вариант №5

В каждом байте с четным количеством битов 1 удалить идущие подряд начальные (старшие) биты 0.

Вариант №6

Удалить из каждого полубайта строки последний (младший) бит при условии, что у этого полубайта не совпадают первый и последний биты.

Вариант №7

Сложить по модулю 2 байты, в которых нет k идущих подряд 0, но есть k идущих подряд 1.

Вариант №8

Если байт начинается с бита 1 (старший бит равен 1), то разбить все его биты на пары, и, если в паре биты одинаковые, заменить их на два бита 0, если разные – на один бит 1.

Вариант №9

Удвоить в каждом байте все биты, совпадающие с первым (старшим) битом байта.

Вариант №10

Если в байте количество битов 1 совпадает с количеством битов 0, то переставить в каждом полубайте этого байта биты в обратном порядке.

Вариант №11

Найти максимальное количество идущих подряд групп по три бита, начинающихся и заканчивающихся на разные биты.

Вариант №12

Если в байте есть два идущих подряд бита 1, то переставить в каждом полубайте этого байта биты 0 и 2, а также 1 и 3.

Вариант №13

Определить количество таких битов 1, что справа и слева стоят биты 0, и количество таких битов 0, что справа и слева стоят биты 1.

Вариант №14

В каждом байте инвертировать k-й бит. После каждого байта, в котором равное количество битов 0 и 1, добавить бит 0.

Вариант №15

В байтах, начинающихся или заканчивающихся на заданный бит, после всех битов 1 добавить еще один бит 1.

Вариант №16

Удалить каждый k-й бит.

Вариант №17

Попарно переставить соседние биты в каждом байте строки. Добавить после каждой пары бит, совпадающий с первым (старшим) битом пары.

Вариант №18

В каждом полубайте, заканчивающемся на заданный бит (младший бит), удалить идущие подряд конечные биты 1.

Вариант №19

Если в полубайте количество битов 0 и 1 одинаковое, оставить от полубайта первый и последний биты, если разное – добавить после полубайта бит с большим количеством.

Вариант №20

В каждом байте, в котором нет трех идущих подряд битов 1, переместить все биты 0 к правому краю.

Вариант №21

Если в байте в каком-либо полубайте количество битов 0 совпадает с количеством битов 1, то добавить перед таким байтом бит 0.

Вариант №22

Добавить к каждому полубайту строки в начало и конец бит, совпадающий с первым (старшим) битом этого полубайта.

Вариант №23

В каждом байте строки циклически сдвинуть биты на k позиций влево. Сложить по модулю 2 все биты строки.

Вариант №24

Если в байте совпадают два первых (старших) бита с двумя последними (младшими) битами, то переставить в этом байте биты в обратном порядке.

Вариант №25

Определить длину максимальной последовательности из 1 в двоичном представлении строки, а также номер байта и номер бита в этом байте начала этой последовательности.

ЛАБОРАТОРНАЯ РАБОТА № 1.2

Задача. *Подсчитать сумму и количество чисел для последовательности, введенной с клавиатуры. Окончание ввода – число 1000.*

```
int main() {
  int a, k = 0, s = 0;
  printf("Введите последовательность чисел\n");
  scanf("%d", &a);      // ввод первого числа до цикла
  while (a != 1000) {   // пока не конец ввода (число не 1000)
    k++;                // обработка очередного числа
    s += a;
    scanf("%d", &a);    // ввод очередного числа в самом конце цикла
  }
  printf("Количество чисел = %d\n", k);
  printf("Сумма      чисел = %d\n", s);
  return 0;
}
```

Задача. *Подсчитать количество чисел, введенных с клавиатуры. Окончание ввода – превышение суммы положительных чисел числа 100.*

```
int main() {
  int a;
  printf("Введите последовательность чисел\n");
  scanf("%d", &a);
  // если первое число положительное, то заносим его в сумму
  // иначе обнуляем сумму
  int s = a > 0 ? a : 0;
  int k = 0;
  while (s <= 100) {
    k++;
    scanf("%d", &a);
    if (a > 0)
      s += a;   // добавляем очередное положительное число в сумму
  }
  printf("Количество чисел = %d\n", k);
  return 0;
}
```

Задача. *Подсчитать количество отрицательных и сумму положительных чисел для последовательности чисел, введенной с клавиатуры. Окончание ввода – число 0.*

```
int main() {
  int a, k = 0, s = 0;
  printf("Введите последовательность чисел\n");
  scanf("%d", &a);
  while (a != 0) {
    if (a < 0)
      k++;
```

```
    else
      s += a;
    scanf("%d", &a);
  }
  printf("Количество отрицательных чисел = %d\n", k);
  printf("Сумма      положительных чисел = %d\n", s);
  return 0;
}
```

Задача. Подсчитать количество пар совпадающих чисел, введенных с клавиатуры. Окончание ввода – число, кратное 5.

```
int main() {
  int a, k = 0, apred;
  printf("Введите последовательность чисел\n");
  scanf("%d", &a);
  apred = 5; // делаем предыдущее таким, каким не может быть число послед.
  while (a%5 != 0) { // есть остаток от деления на 5
    if (a == apred)  // если очередное число совпадает с предыдущим
      k++;           // количество совпадающих пар увеличиваем на 1
    apred = a;       // сохраняем предыдущее введенное число
    scanf("%d", &a);
  }
  printf("Количество пар совпадающих чисел = %d\n", k);
  return 0;
}
```

Задача. Определить, сколько в последовательности чисел, совпадающих с первым числом последовательности. Окончание ввода – отрицательное число.

```
int main() {
  int a, a1, k = 0;
  printf("Введите последовательность чисел\n");
  scanf("%d", &a);
  a1 = a;  // запоминаем первое число последовательности
  while (a >= 0) {
    if (a == a1)
      k++;
    scanf("%d", &a);
  }
  printf("С первым членом совпадает %d чисел\n", k);
  return 0;
}
```

Задача. Определить, есть ли в последовательности два нечётных числа подряд. Окончание ввода – число 0 или число 100.

```
int main() {
  int a;
  printf("Введите последовательность чисел\n");
  scanf("%d", &a);
```

```
  int apred = 0;   // для первого числа считаем предыдущее число чётным
  int p = 0;       // считаем сначала, что двух нечётных чисел подряд нет
  while (a != 0 && a != 100) {
    if (a%2 == 1 && apred%2 == 1)
      p = 1;    // устанавливаем признак, что есть два нечётных числа подряд
    apred = a; // сохраняем предыдущее число
    scanf("%d", &a);
  }
  if (p == 0)
    printf("Нет двух нечётных чисел подряд\n");
  else
    printf("Есть два нечётных числа подряд\n");
  return 0;
}
```

ЛАБОРАТОРНАЯ РАБОТА № 1.3

Задача. *Ввести число и вывести все его цифры в обратном порядке по одной в строке.*

```
int main() {
  unsigned long x;
  int c;                // цифра числа
  scanf("%lu", &x);     // вводим число
  while (x != 0) {      // пока у числа есть цифры; можно и так: while (x)
    c = x % 10;         // получаем последнюю цифру числа
    printf("c = %d\n", c); // выводим последнюю цифру числа
    x /= 10;            // убираем последнюю цифру числа
  }
  return 0;
}
```

Задача. *Сформировать число по его цифрам. Окончание ввода – цифра 9.*

```
int main() {
  int c;                // цифра числа
  unsigned long x = 0;  // начальное значение для числа равно 0
  scanf("%d", &c);      // вводим первую цифру числа
  while (c != 9) {      // пока введена не цифра 9
    x = x*10 + c;       // добавляем цифру в конец числа
    scanf("%d", &c);    // вводим очередную цифру числа
  }
  printf("x = %lu\n", x);
  return 0;
}
```

Задача. *Переставить цифры в числе в обратном порядке (125 => 521).*

```
int main() {
  unsigned long x, y = 0;
  int c;
  scanf("%lu", &x);
  while (x) {           // пока у числа есть цифры
    c = x % 10;         // получаем последнюю цифру числа
    y = y*10 + c;       // добавляем цифру в конец нового числа
    x /= 10;            // убираем последнюю цифру числа
  }
  printf("y = %lu\n", y);
  return 0;
}
```

Задача. *Удалить старшую цифру числа (725 => 25).*

```
int main() {
  unsigned long x, y = 0, p = 1;
  scanf("%lu", &x);
  while (x > 9) {      // пока больше одной цифры в числе
    y += (x%10 * p);   // добавляем цифру в начало нового числа
    x /= 10;           // убираем последнюю цифру числа
    p *= 10;
  }
  printf("y = %lu\n", y);
  return 0;
}
```

Второй вариант:

```
// x = 12345
// k = 10000
// y = 12345 % 10000 = 2345
int main() {
  unsigned long x, y, xx, k = 1;
  scanf("%lu", &x);
  xx = x;              // сохраняем число
  while (x > 9) {      // пока больше одной цифры в числе
    k *= 10;
    x /= 10;           // убираем последнюю цифру числа
  }
  y = xx % k;          // удаляем первую цифру числа
  printf("y = %lu\n", y);
  return 0;
}
```

ЛАБОРАТОРНАЯ РАБОТА № 1.4

Задача. *Вычислить значение функции на интервале [a;b] в n равноудаленных точках. Обеспечить контроль введенных и полученных данных.*

$$f=\begin{cases}\cos(\pi x)+\dfrac{7-x^2}{|x|}, & x<2,\\ \ln(x-u)-\dfrac{\sqrt{x}}{u}, & x\ge 2.\end{cases}$$

```
#include <math.h>                  // для математических функций
int main(){
  int n, err;
  float a, b, u, x, f, h;
  printf("Введите a, b, u, n\n");
  scanf("%f %f %f %d", &a, &b, &u, &n);
  if (n < 2)
    n = 10;                        // количество точек по умолчанию
  if (a == b) {
    a -= 0.5;                      // расширение интервала
    b += 0.5;
  }
  else
    if (a > b) {
      x = a;                       // перестановка концов интервала
      a = b;
      b = x;
    }
  h = (b - a) / (n - 1);
  for (x = a; x < b+0.5*h; x += h){
    err = 0;  // сначала считаем, что функция определена в точке x
    if (x < 2) {
      if (x == 0) // деление на 0
        err = 1;   // функция не определена в точке x
      else
        f = cos(M_PI*x)+(7-pow(x,2)) / fabs(x); // pow(x, 2) = x*x
    }
    else {
      if (u == 0 || x-u <= 0)   // деление на 0 или
                                // логарифм не положительного числа
        err = 1;   // функция не определена в точке x
      else
        f = log(x-u)-pow(x,0.5)/u; // pow(x, 0.5) = sqrt(x)
    }
    printf("|%10.4f  |", x);
    if (err != 0)
      printf("  не определена!  |\n");
    else
```

```
        printf("%16.4f  |\n", f);
    }
    return 0;
}
```

Стандартные математические функции:

В любых арифметических выражениях можно использовать стандартные математические функции, которые можно применять к любым числовым операндам.

При использовании этих функций в программу необходимо включить файл <math.h> (все параметры имеют тип double):

`sin(x)` – синус
`cos(x)` – косинус
`tan(x)` – тангенс
`exp(x)` – экспонента e^x
`log(x)` – натуральный логарифм ln(x)
`log10(x)` – десятичный логарифм $\log_{10}(x)$
`sqrt(x)` – квадратный корень
`pow(x, y)` – возведение в степень x^y
`fabs(x)` – абсолютная величина для double

Также можно пользоваться макроопределением для константы π, которое тоже находится в файле <math.h>:

```
#define M_PI   3.14159265358979323846
```

При использовании этих функций в программу необходимо включить файл <stdlib.h>:

`abs(x)` – абсолютная величина для параметра типа int
`labs(x)` – абсолютная величина для параметра типа long

ЛАБОРАТОРНАЯ РАБОТА № 2.1

Задача. *Найти сумму элементов одномерного массива.*

```
int main() {
  int a[100], n, i, n_max;
  int s = 0;
  n_max = 100;          // n_max = sizeof(a) / sizeof(int);
  // ввод размерности массива с проверкой на правильность
  // зацикливание, пока не будет введена правильная размерность
  do {
    printf("Введите количество элементов массива n = ");
    scanf("%d", &n);
  } while (n < 0 || n > n_max);
  printf("Введите элементы массива\n");
  for (i = 0; i < n; i++) {
    printf("a[%d] = ", i);
    scanf("%d", &a[i]);
  }
  printf("Вы ввели массив\n");
  for (i = 0; i < n; i++)
    printf("%d ", a[i]);
  printf("\n");
  for (i = 0; i < n; i++)
    s += a[i];
  printf("Сумма элементов массива = %d\n", s);
  return 0;
}
```

Задача. *Отсортировать одномерный массив по возрастанию.*

```
int main(void) {
  int a[100], n;
  int i, j, b;
  scanf("%d", &n);
  for (i = 0; i < n; i++)
    scanf("%d", &a[i]);
  // сортировка – это всегда два вложенных цикла
  for (i = 0; i < n - 1; i++)     // каждый элемент кроме последнего
    for (j = i + 1; j < n; j++)   // сравниваем со всеми после него
      if (a[i] > a[j]) {
        b = a[i];                 // перестановка элементов
        a[i] = a[j];
        a[j] = b;
      }
  for (i = 0; i < n; i++)
    printf("%d ", a[i]);
  printf("\n");
  return 0;
}
```

<u>Задача.</u> Сформировать массив B из чётных элементов массива A[n].

```
int main() {
  int a[100], b[100], na, nb, i, j;
  printf("Введите количество элементов в массиве A: ");
  scanf("%d", &na);
  printf("Введите элементы массива A\n");
  for (i = 0; i < na; i++)
    scanf("%d", &a[i]);
  j = 0;   // индекс первого формируемого элемента массива B равен 0
  for (i = 0; i < na; i++)
    if (a[i]%2 == 1) {    // if (!(a[i] & 1))
      b[j] = a[i]; // формирование массива B
      j++;         // индекс следующего формируемого элемента массива B
    }
  nb = j;          // количество элементов в сформированном массиве B
  printf("Сформирован массив B\n");
  for (j = 0; j < nb; j++)
    printf("%d ", b[j]);
  return 0;
}
```

ЛАБОРАТОРНАЯ РАБОТА № 2.2

Задача. *Найти сумму элементов массива A[n][m]. В заданной строке найти максимальный элемент.*

```
int main() {
  int a[10][20], n, m, i, j, k;
  int s = 0, max;
  printf("Введите размерность массива n и m\n");
  scanf("%d %d", &n, &m);
  printf("Введите элементы массива\n");
  for (i = 0; i < n; i++)
    for (j = 0; j < m; j++)
      scanf("%d", &a[i][j]);
  printf("Вы ввели массив\n");
  for (i = 0; i < n; i++) {
    for (j = 0; j < m; j++)
      printf ("%5d ", a[i][j]);
    printf("\n");
  }
  // поиск суммы элементов массива A[n][m]
  for (i = 0; i < n; i++)
    for (j = 0; j < m; j++)
      s += a[i][j];
  printf("Сумма элементов массива = %d\n", s);
  // поиск максимального элемента в строке с индексом k
  printf("Введите индекс строки k = ");
  scanf("%d", &k);
  // считаем максимальным первый элемент строки (с индексом 0)
  max = a[k][0];
  for (j = 1; j < m; j++)   // перебираем элементы строки
    if (a[k][j] > max)   // если очередной элемент больше максимального
      max = a[k][j];     // считаем максимальным этот элемент строки
  printf("Максимальный элемент в %d строке = %d\n", k, max);
  return 0;
}
```

Обратить внимание! В двумерном массиве строка, столбец и диагонали являются по своей сути одномерными массивами. Поэтому при обработке строки, столбца или диагоналей достаточно использовать один цикл, а не два вложенных.

ЛАБОРАТОРНАЯ РАБОТА № 3.1

Задача. *Используя указатели, найти максимальное из двух чисел. Затем увеличить первое число на 1, а второе – умножить на 2.*

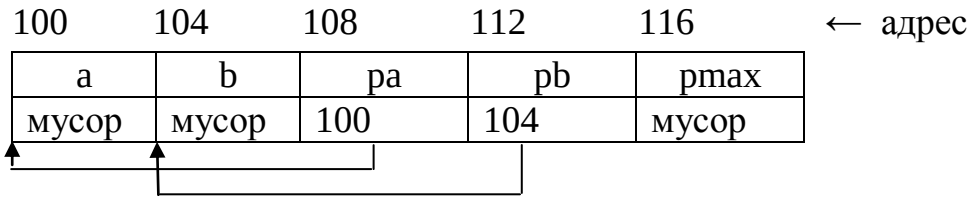
```
int main() {
  int a, b, max;
  int *pa, *pb, *pmax;   // описываем три указателя
  pa = &a;       // направляем указатель pa на переменную a
  pb = &b;       // направляем указатель pb на переменную b
  pmax = &max;   // направляем указатель pmax на переменную max
  printf("Введите a и b");
  scanf("%d %d", pa, pb);    // при вводе нужны адреса переменных
  // туда, куда указывает pmax, заносим максимальное среди значений,
  // на которые указывают pa и pb
  *pmax = *pa > *pb ? *pa : *pb;
  (*pa)++;    // к тому, на что указывает pa, добавляем 1 (приоритет операций!)
  *pb *= 2;   // то, на что указывает pb, умножаем на 2
  printf("Максимальное число = %d\n", *pmax);
  printf("a = %d\n", *pa);   // при выводе нужны значения переменных
  printf("b = %d\n", *pb);   // при выводе нужны значения переменных
  return 0;
}
```

или так можно (без переменной для максимума):

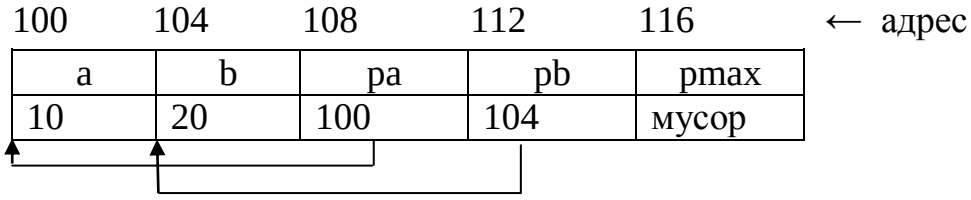
```
int main() {
  int a, b;
  int *pa, *pb, *pmax;   // описываем три указателя
  pa = &a;   // направляем указатель pa на переменную a
  pb = &b;   // направляем указатель pb на переменную b
  // рисунок после выполнения этих операторов (1)
  printf("Введите a и b");
  scanf("%d %d", pa, pb);    // при вводе нужны адреса переменных
  // рисунок после выполнения этих операторов (вводим 10 и 20) (2)
  pmax = *pa > *pb ? pa : pb; // pmax указывает на максимальное число
  // рисунок после выполнения этого оператора (3)
  printf("Максимальное число = %d\n", *pmax);
  // на экране будет "Максимальное число = 20"
  (*pa)++;    // к тому, на что указывает pa, добавляем 1
  *pb *= 2;   // то, на что указывает pb, умножаем на 2
  // рисунок после выполнения этих операторов (4)
  printf("a = %d\n", *pa);    // при выводе нужны значения переменных
  // на экране будет "a = 11"
  printf("b = %d\n", *pb);    // при выводе нужны значения переменных
  // на экране будет "b = 40"
  return 0;
}
```

Рисунки для второго решения (предполагаем, что переменные располагаются в памяти друг за другом по мере их описания, начиная с адреса 100):

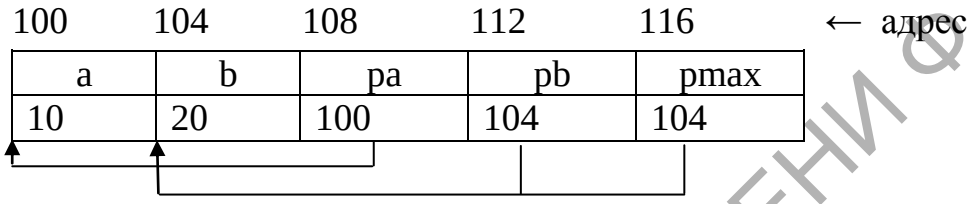
(1)



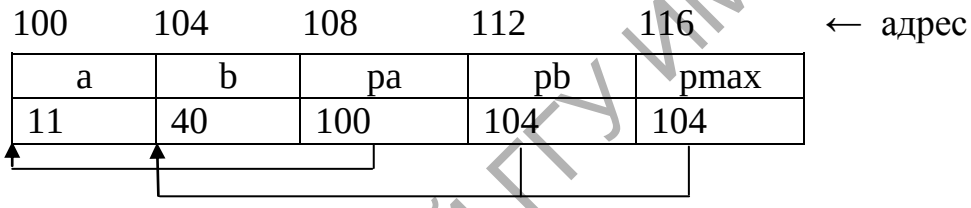
(2)



(3)



(4)



ЛАБОРАТОРНАЯ РАБОТА № 3.2

Задача. Найти сумму элементов массива A[n]. Для доступа к элементам массива использовать указатели.

```
int main() {
  int a[100], n, i;
  int s = 0;
  printf("Введите количество элементов массива n = ");
  scanf("%d", &n);
  printf("Введите элементы массива\n");
  for (i = 0; i < n; i++)
    scanf("%d", a + i);   // a + i → адрес i-го элемента массива a, т.е. &a[i]
  printf("Вы ввели массив\n");
  // *(a + i) → значение i-го элемента массива a, т.е. a[i]
  for (i = 0; i < n; i++)
    printf("%d ", *(a + i));
  printf("\n");
  for (i = 0; i < n; i++)
    s += *(a + i);   // *(a + i) → значение i-го элемента массива a, т.е. a[i]
  printf("Сумма элементов массива = %d\n", s);
  return 0;
}
```

```
==== второй вариант ================================
```

```
int main() {
  int *pa;
  int a[100], n, i;
  int s = 0;
  printf("Введите количество элементов массива n = ");
  scanf("%d", &n);
  printf("Введите элементы массива\n");
  // a → адрес начала массива a, т.е. &a[0]
  // pa = a → направляем pa на начало массива a
  // pa++ → переходим к следующему элементу массива a
  for (pa = a, i = 0; i < n; i++, pa++)
    scanf("%d", pa);// pa → адрес очередного элемента массива a, т.е. &a[i]
  printf("Вы ввели массив\n");
  // *pa → значение очередного элемента массива a, т.е. a[i]
  for (pa = a, i = 0; i < n; i++, pa++)
    printf("%d ", *pa);
  printf("\n");
  for (pa = a, i = 0; i < n; i++, pa++)
    s += *pa;   // *pa → значение очередного элемента массива a, т.е. a[i]
  printf("Сумма элементов массива = %d\n", s);
  return 0;
}
```

==== третий вариант ====================================

```
int main() {
  int *pa, *pa_n;
  int a[100], n;
  int s = 0;
  printf("Введите количество элементов массива n = ");
  scanf("%d", &n);
  pa_n = a + n; // направляем pa_n на конец последнего элемента массива a
  printf("Введите элементы массива\n");
  // a → адрес начала массива a
  // pa = a → направляем pa на начало массива a
  // pa < pa_n → пока pa_n указывает на элемент массива a
  // (не дошли до адреса конца последнего элемента массива)
  // pa++ → переходим к следующему элементу массива a
  for (pa = a; pa < pa_n; pa++)
    scanf("%d", pa);
  printf("Вы ввели массив\n");
  for (pa = a; pa < pa_n; pa++)
    printf("%d ", *pa);
  printf("\n");
  for (pa = a; pa < pa_n; pa++)
    s += *pa;
  printf("Сумма элементов массива = %d\n", s);
  return 0;
}
```

ЛАБОРАТОРНАЯ РАБОТА № 3.3

<u>Задача.</u> Найти максимальные элементы в массиве A[n][n] и на его главной диагонали. Для доступа к элементам массива использовать указатели.

```
int main() {
  int a[10][10], n, i, j;
  int max, max_d;
  printf("Введите размерность массива n = ");
  scanf("%d", &n);
  printf("Введите массив\n");
  // *(a + i) → адрес начала i-ой строки массива a
  // *(a + i) + j → адрес начала j-го элемента i-ой строки массива a, т.е. &a[i][j]
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      scanf("%d", *(a + i) + j);
  printf("Вы ввели массив\n");
  // *(*(a + i) + j) → значение j-го элемента i-ой строки массива a, т.е. a[i][j]
  for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++)
      printf("%4d ", *(*(a + i) + j));
    printf("\n");
  }
  // поиск максимального элемента в массиве a
  max = **a;   // максимальным считаем элемент a[0][0]
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      // если очередной элемент a[i][j] больше максимального
      if (max < *(*(a + i) + j))
        max = *(*(a + i) + j);   // максимальным считаем a[i][j]
  printf("Максимальный элемент массива = %d\n", max);
  // поиск максимального элемента на главной диагонали массива a
  max_d = **a;   // максимальным считаем элемент a[0][0]
  // главная диагональ – одномерный массив, поэтому один цикл
  for (i = 0; i < n; i++)
    // если очередной элемент a[i][i] больше максимального
    if (max_d < *(*(a + i) + i))
      max_d = *(*(a + i) + i);   // максимальным считаем a[i][i]
  printf("Максимум на диагонали массива = %d\n", max_d);
  return 0;
}
```

ЛАБОРАТОРНАЯ РАБОТА № 4.1

Задача. Подсчитать длину строки.

```
int main(){
  char s[80];
  int n = 0;
  gets(s);
  // пока не встретим символ '\0'
  while (s[n] != 0)   // while (s[n]) или  while (s[n] != '\0')
    n++;
  printf("n = %d\n", n);
  return 0;
}
```

Задача. Подсчитать длину строки, используя указатель.

```
int main(){
  char s[80], *ps;
  int n = 0;
  gets(s);
  ps = s;          // направляем ps на начало строки s
  while (*ps) {    // пока символ, на который указывает ps, не равен '\0'
    n++;
    ps++;          // переходим к следующему символу строки s
  }
  printf("n = %d\n", n);
  return 0;
}
=======  второй вариант ============================
int main(){
  char s[80], *ps;
  gets(s);
  ps = s;          // направляем ps на начало строки s
  while (*ps)      // пока символ, на который указывает ps, не равен '\0'
    ps++;          // переходим к следующему символу строки s
  // сейчас ps указывает на '\0'
  // ps – s → количество символов между двумя указателями
  printf("n = %d\n", ps - s);
  return 0;
}
```

Задача. Сколько цифр в строке?

```
int main(){
  char s[80];
  int i = 0, n = 0;
  gets(s);
  while (s[i] != '\0') {
    if (s[i] >= '0' && s[i] <= '9') // символ цифра?
```

```
      n++;
    i++;
  }
  printf("В слове %s цифр %d\n", s, n);
  return 0;
}
```

Задача. К строке 1 присоединить строку 2.

```
int main(){
  char s1[80],s2[80];
  int i = 0, j = 0;
  gets(s1);
  gets(s2);
  while (s1[i])        // становимся в конец строки s1
    i++;
  while (s2[j]) {      // добавляем к s1 строку s2
    s1[i] = s2[j];     // s1[i++] = s2[j++];
    i++;               //
    j++;               //
  }
  s1[i] = 0;   // не забываем ставить символ конца строки!!!
  puts(s1);
  return 0;
}
=======  второй вариант ============================
int main(){
  char s1[80],s2[80];
  int i,j;
  gets(s1);
  gets(s2);
  for (i = 0; s1[i]; i++);        // становимся в конец строки s1
  for (j = 0; s2[j]; i++, j++)    // добавляем к s1 строку s2
    s1[i] = s2[j];
  s1[i] = 0;   // не забываем ставить символ конца строки!!!
  puts(s1);
  return 0;
}
```

Задача. К строке 1 присоединить строку 2, используя указатели.

```
int main(){
  char s1[80],s2[80];
  char *ps1,*ps2;
  gets(s1);
  gets(s2);
  ps1 = s1;   // направляем ps1 на начало строки s1
  ps2 = s2;   // направляем ps2 на начало строки s2
```

```
  while (*ps1)      // двигаемся по строке s1 с помощью указателя ps1
    ps1++;
  while (*ps2) {    // двигаемся по строке s2 с помощью указателя ps2
    *ps1 = *ps2;
    ps1++;
    ps2++;
  }
  *ps1 = 0;   // не забываем ставить символ конца строки!!!
  puts(s1);
  return 0;
}
```

Задача. *Удалить из строки все вхождения заданного символа.*

```
int main() {
  char s[80], ss[80], c;
  int i = 0;   // индекс для просмотра исходной строки s
  int j = 0;   // индекс для формирования новой строки ss
  gets(s);
  c = 'a';              // символ, который будем удалять
  while (s[i]) {
    if (s[i] != c) {
      ss[j] = s[i];     // создаем новую строку
      j++;              // индекс формируемой строки
    }
    i++;
  }
  ss[j] = 0;   // в новую строку ставим признак конца строки
  puts(ss);
  return 0;
}
=======  второй вариант ============================
int main() {
  char s[80], c;
  int i = 0;   // индекс для просмотра исходной строки s
  int j = 0;   // индекс для формирования нового содержимого строки s
  gets(s);
  c = 'a';
  while (s[i]) {
    if (s[i] != c) {
      // формируем новое содержимое прямо в исходной строке s
      s[j] = s[i];
      j++;
    }
    i++;
  }
  s[j] = 0;   // ставим признак конца строки в конец нового содержимого s
  puts(s);
```

```
  return 0;
}
=======  третий вариант ============================
int main() {
  char s[80], c;
  char *ptr;     // указатель для просмотра исходного содержимого строки s
  char *ptr_rez;// указатель для формирования нового содержимого строки s
  ptr = ptr_rez = s;   // направляем оба указателя на начало строки s
  gets(s);
  c = 'a';
  while (*ptr) {
    if (*ptr != c) {
      *ptr_rez = *ptr;
      ptr_rez++;
    }
    ptr++;
  }
  *ptr_rez = 0;
  puts(s);
  return 0;
}
```

Задача. *Дана строка. Создать подстроку длиной n > 0, начиная от символа с индексом k >= 0.*

```
int main(){
  char s[80], ss[80];
  int k, n, i;
  gets(s);
  scanf("%d%d", &n, &k);
  for (i = 0; i < k; i++)   // пропускаем k символов строки s
    if (!s[i])     // если символ '\0' встречаем раньше
      break;       // выходим из цикла
  if (i < k)       // если длина строки меньше чем k
    ss[0] = 0;   // ответ – пустая строка
  else {
    // копируем в строку ss n символов из строки s
    // или все символы до конца строки, если после k-го символа нет n символов
    for (i = 0; i < n && s[i+k]; i++)
      ss[i] = s[i+k];
    ss[i] = 0;   // в новую строку ставим признак конца строки
  }
  puts(ss);
  return 0;
}
```

<u>Задача.</u> Дана строка не нулевой длины. Добавить в начало строки n *раз последний символ строки и в конец строки* n *раз первый символ строки.*

```
int main(){
  char s[80], c_end;
  int i = 0, j, n;
  gets(s);
  scanf("%d", &n);
  while (s[i])          // становимся на конец строки s
    i++;
  c_end = s[i-1];       // запоминаем последний символ строки s
  for (j = 0; j < n; j++)   // добавляем в конец n первых символов
    s[i+j] = s[0];
  s[i+j] = 0;           // ставим признак конца строки
  j += i;
  for (i = j; i >= 0; i--) // смещаем символы вправо на n позиций
    s[i+n] = s[i];
  for (i = 0; i < n; i++)   // в первые n символов записываем
    s[i] = c_end;           // последний символ
  puts(s);
  return 0;
}
```

ЛАБОРАТОРНАЯ РАБОТА № 4.2

Задача. *Дано предложение, слова в котором разделены ровно одним пробелом. Найти количество слов в предложении.*

```
int main(){
  char s[80];
  gets(s);
  // если предложение не пустое, то одно слово точно есть
  int n = s[0] ? 1 : 0;
  int i = 0;
  while (s[i]) {
    if (s[i] == ' ')
      n++; // за пробелом есть очередное слово
    i++;
  }
  printf("n = %d\n", n);
}
```

Задача. *Дано предложение, слова в котором разделены ровно одним пробелом. Оставить от каждого слова только первую букву, поставив после нее длину слова ("волк" => "в4").*

```
#include <stdlib.h>      // для функции itoa()
int main(){
  char s[80], ss[80];
  char t[3];      // для преобразования числа в строку
  int i, j = 0, n, k;
  gets(s);
  for (i = 0; s[i] != 0; i++) {
    // начало очередного слова
    ss[j] = s[i]; // копируем первый символ слова
    j++;
    n = i;
    // проходим все символы слова (до пробела или до конца предложения
    while (s[i] != ' ' && s[i])
      i++;
    n = i - n;          // находим длину слова
    itoa(n, t, 10);   // преобразуем длину слова в строку
    // добавляем строку с длиной слова после первого символа слова
    for (k = 0; t[k]; ss[j++] = t[k++]);
    if (s[i] == 0)   // если после слова символ '\0'
      break;          // выходим из цикла (больше слов в предложении нет)
    ss[j++] = ' ';   // ставим пробел после очередного слова
  }
  ss[j] = 0;   // добавляем в предложение символ '\0'
  puts(ss);
  return 0;
}
```

Задача. *Дано предложение, слова в котором разделены ровно одним пробелом. Сформировать массив слов из слов предложения.*

```
int main() {
  int n = 0, i, j;
  char s[80], w[10][80];
  gets(s);
  i = 0;
  while (s[i] != 0) {
    // начало очередного слова
    j = 0;
    // после слова будет пробел или символ '\0'
    while (s[i] != ' ' && s[i] != 0) {   // копируем слово
      w[n][j] = s[i];
      j++;
      i++;
    }
    w[n][j] = 0;      // добавляем в слово символ '\0'
    n++;              // увеличиваем счётчик слов
    if (s[i] == 0)    // если после слова символ '\0'
      break;          // выходим из цикла (больше слов в предложении нет)
    i++;
  }
  for (i = 0; i < n; i++)
    puts(w[i]);
  return 0;
}
=======  второй вариант ============================
int main(void){
  char s[80], w[20][80];
  int i = 0, j, n = 0;
  gets(s);
  while (s[i]){
    for (j = 0; s[i] != ' ' && s[i]; w[n][j++] = s[i++]);
    w[n++][j] = 0;
    if (s[i] == 0)
      break;
    i++;
  }
  for (i = 0; i < n; i++)
    puts(w[i]);
  return 0;
}
```

Задача. *Дано предложение, слова в котором разделены ровно одним пробелом. Записать все его слова через черточку, удалив из предложения слова, в которых есть цифры.*

```
int main() {
  char sIn[80], sOut[80];
  int i, j, k;
  int iBeg;
  int prDigit;
  puts("Введите предложение");
  gets(sIn);
  i = 0;
  j = 0;
  while (sIn[i] != 0) {
    // начало очередного слова
    iBeg = i;           // запоминаем индекс начала слова
    prDigit = 0;        // считаем, что в слове цифр нет
    // проходим слово до конца и ищем цифры
    while (sIn[i] != ' ' && sIn[i] != 0) {
      if (sIn[i] >= '0' && sIn[i] <= '9')
        prDigit = 1;    // устанавливаем признак, что цифры есть
      i++;
    }
    if (prDigit == 0) {  // если цифр в слове нет
      // переносим слово в новое предложение
      for (k = iBeg; k < i; k++) {
        sOut[j] = sIn[k];
        j++;
      }
      sOut[j] = '-';  // после слова ставим черточку
      j++;
    }
    if (sIn[i] == 0)   // выходим из цикла, если конец предложения
      break;
    i++;
  }
  sOut[j] = 0;      // помечаем конец сформированного предложения
  if (j > 0)        // если результат не пустой
    sOut[j-1]=0;    // убираем черточку после последнего слова
  puts("Полученное предложение");
  puts(sOut);
  return 0;
}
```

<u>Задача.</u> Дано предложение, слова в котором разделены ровно одним пробелом. Удалить из предложения слова длиной 2 и 3 символа.

```
int main() {
  char s_in[80], s_out[80];
  int i, j, beg, len;
  puts("Введите предложение");
  gets(s_in);
  i = j = 0;
  while (s_in[i]!=0) {
    // начало очередного слова
    beg = j;   // запоминаем индекс начала нового слова в s_out
    // переносим слово в новое предложение
    while (s_in[i] != ' ' && s_in[i] != 0) {
      s_out[j] = s_in[i];
      j++;
      i++;
    }
    len = j - beg;      // вычисляем длину слова
    if (len == 2 || len == 3)   // слово надо удалять из предложения
      j = beg;          // удаляем слово из нового предложения
    else {
      s_out[j] = ' ';
      j++;
    }
    if (s_in[i] == 0)   // выходим из цикла, если конец предложения
      break;
    i++;
  }
  s_out[j] = 0;     // помечаем конец сформированного предложения
  if (j > 0)        // если результат не пустой
    s_out[j-1]=0;   // убираем пробел после последнего слова
  puts("Полученное предложение");
  puts(s_out);
  return 0;
}
```

ЛАБОРАТОРНАЯ РАБОТА № 4.3

Задача. *Ввести и вывести массив строк. Конец ввода – пустая строка.*

```
int main() {
  int n = 0, i;
  char s[10][80];
  gets(s[n]);
  while (s[n][0] != 0) {     // *s[n] != 0
    n++;
    gets(s[n]);
  }
  for (i = 0; i < n; i++)
    puts(s[i]);
  return 0;
}
```

```
=======  второй вариант ============================
```

```
int main() {
  int n = 0, i;
  char s[10][80];
  while (*gets(s[n]))   // gets() возвращает адрес строки s[n]
    n++;
  for (i = 0; i < n; i++)
    puts(*(s+i));       // адрес строки содержится в s[i]
  return 0;
}
```

Задача. *Создать предложение из слов, вводимых с клавиатуры. Окончание ввода – пустая строка.*

```
int main(void) {
  char w[10], s[120];
  *s = 0;
  while (*gets(w)) {
    strcat(s, w);           // одним оператором:
    strcat(s, " ");         // strcat(strcat(s, w), " ");
  }
  if (*s)                   // предложение не пустое
    *(s+strlen(s)-1) = 0;   // удаляем последний лишний пробел
  puts(s);
  return 0;
}
```

Задача. *Если это возможно, то удалить в строке два первых и два последних символа.*

```
int main(void) {
  char s[80], len;
  gets(s);
  len = strlen(s);
```

```
  if (len >= 4) {
    s[len-2] = 0;      // удаление двух последних символов
    strcpy(s, s+2);    // удаление двух первых символов
  }
  puts(s);
  return 0;
}
```

Задача. Ввести две строки и удалить в первой строке последнее вхождение второй строки.

```
int main(void) {
  char s1[80], s2[80], *p, *pEnd = NULL;
  gets(s1);
  gets(s2);
  p = strstr(s1, s2);      // поиск первого вхождения s2
  while (p != NULL) {      // нашли s2
    pEnd = p;              // сохраняем адрес найденной s2
    p = strstr(p+1, s2);   // поиск следующей подстроки
  }
  if (pEnd != NULL)        // нашли s2 в s1
    strcpy(pEnd, pEnd+strlen(s2));  // удаляем последнюю s2
  puts(s1);
  return 0;
}
```

Задача. Дано предложение. Занести все его слова в массив строк.

```
int main(void){
  char s[120], w[20][40], *p;
  int k = 0, i = 0;
  gets(s);
  p = strtok(s, " ");
  while (p != NULL) {
    strcpy(w[k++], p);
    p = strtok(NULL, " ");
  }
  while (i < k)
    puts(w[i++]);
  return 0;
}
```

Задача. Дано предложение. Создать новое предложение, удалив из исходного знаки пунктуации и лишние пробелы.

```
int main(void){
  char s1[120], s2[120] = "", *p, d[] = " .,:;!?-";
  gets(s1);
  p = strtok(s1, d);     // выделяем первое слово предложения s1
  while (p != NULL) {    // пока удалось выделить следующее слово в s1
```

```
    // добавляем выделенное слово в конец s2, а следом добавляем пробел
    strcat(strcat(s2, p), " ");
    p = strtok(NULL, d);    // выделяем следующее слово в s1
  }
  if (s2[0])                // предложение не пустое (в нём есть слова)
    s2[strlen(s2)-1] = 0;   // удаляем последний лишний пробел
  puts(s2);
  return 0;
}
```

Задача. *Дано предложение. Создать массив указателей на встречающиеся в нем слова.*

```
int main(void) {
  char s[80], *p;
  char *d[60];  // массив указателей на слова предложения
  int k = 0;    // количество заполненных указателей в массиве
  int i;
  gets(s);
  p = strtok(s), " ");  // p = адрес первого слова
  while (p != NULL) {   // слово есть
    for (i = 0; i < k; i++)  // ищем слово в массиве
      if (!strcmp(p, d[i]))  // если нашли слово в массиве
        break;               // выходим из цикла поиска
    if (i == k)       // если слово еще не встречалось
      d[k++] = p;     // сохраняем его адрес в массиве указателей
    p = strtok(NULL, " ");    // p = адрес следующего слова
  }
  for (i = 0; i < k; puts(w[i++]));
  return 0;
}
```

Задача. *Дано предложение. Реверсировать каждое его слово.*

```
int main(void) {
  char s[120], ss[120], *p, lenp;
  gets(s);
  strcpy(ss, s);
  p = strtok(s, " ");  // p = адрес первого слова
  while (p != NULL) {  // слово есть
    lenp = strlen(strrev(p));   // реверсируем слово
    p[lenp] = ss[p-s+lenp]; // убираем поставленный strtor() символ '\0'
    p = strtok(NULL, " ");  // p = адрес следующего слова
  }
  puts(s);
  return 0;
}
```

ЛАБОРАТОРНАЯ РАБОТА № 5.1

Задача. *Написать две функции для вычисления суммы двух отрицательных чисел и их вызов из функции main(). Исходные данные должны вводиться в функции main(). Первая функция должна возвращать заданную величину. Во второй функции обеспечить контроль правильности исходных данных. Функция, кроме вычисления заданной величины, должна возвращать признак правильности исходных данных.*

```
int sum1(int a, int b) {
  return a + b;
}

int sum2(int a, int b, int *sum) {
  if (a >= 0 || b >= 0)
    return 0;     // возврат признака неверных данных
  *sum = a + b;
  return 1;       // возврат признака правильных данных
}

// вызываем функции и сразу используем то, что они вернули
// предварительно возвращённые значения не сохраняем
int main(void) {
  int x, y, s;
  scanf("%d %d", &x, &y);
  printf("Сумма 1 = %d\n", sum1(x, y));
  if (sum2(x, y, &s) == 1)
    printf("Сумма 2 = %d\n", s);
  else
    printf("Неверные данные!\n");
  return 0;
}
```

Второй вариант функции main():

```
// сначала вызываем функции и сохраняем то, что они вернули
// потом используем эти сохранённые значения
int main() {
  int x, y, s1, s2;
  scanf("%d %d", &x, &y);
  s1 = sum1(x, y);
  printf("Сумма 1 = %d\n", s1);
  int flag = sum2(x, y, &s2);
  if (flag == 1)
    printf("Сумма 2 = %d\n", s2);
  else
    printf("Неверные данные!\n");
  return 0;
}
```

ЛАБОРАТОРНАЯ РАБОТА № 5.3

Задача. *Ввести из файла произвольное количество целых чисел и вывести их на экран.*

```
int main() {
  int t;
  char fname[80];
  puts("Введите имя файла");
  gets(fname);
  FILE *pf = fopen(fname, "rt");
  if (pf == NULL) {
    printf("Ошибка открытия файла %s!\n", fname);
    return 0;
  }
  printf("Числа из файла:");
  fscanf(pf, "%d", &t);    // ввод из файла первого числа до цикла
  while (feof(pf) == 0) {  // заходим в цикл, если число введено
    printf(" %d", t);      // обработка введённого числа
    fscanf(pf, "%d", &t); // ввод из файла очередного числа
                           // в самом конце цикла
  }
  printf("\n");
  fclose(pf);
  return 0;
}
```

Задача. *Ввести из файла размерность одномерного массива и его элементы. Вывести введенный массив на экран.*

```
int main() {
  int i, n;
  int mas[80];
  char fname[80] = "in.txt";
  FILE *pf = fopen(fname, "rt");
  if (pf == NULL) {
    printf("Ошибка открытия файла %s!\n", fname);
    return 0;
  }
  fscanf(pf, "%d", &n);
  for (i = 0; i < n; i++) {
    fscanf(pf, "%d", &mas[i]);
  }
  fclose(pf);
  printf("Размерность массива: %d\n", n);
  printf("Массив:");
  for (i = 0; i < n; i++)
    printf(" %d", mas[i]);
  printf("\n");
  return 0;
}
```

Задача. *Разработать функцию для нахождения максимального элемента в одномерном массиве целых чисел.*

```
int fmax(int *m, int n) {
  int max, i;
  max = m[0];
  for (i = 1; i < n; i++)
    if (m[i] > max)
      max = m[i];
  return max;
}

int main() {
  int mas[100], n, i, n2, k, *p, max2;
  printf("Введите размерность массива\n");
  scanf("%d", &n);
  printf("Введите массив\n");
  for (i = 0; i < n; i++)
    scanf("%d", &mas[i]);
  // находим максимум в массиве
  printf("Максимальный элемент = %d\n", fmax(mas, n));
  // находим максимум во второй половине массива
  k = n2 = n / 2;   // k – смещение второй половины
  if (n & 1)   // n2 – количество элементов во второй половине
    n2++;      // при нечетном n средний элемент относим ко второй половине
  printf("Максимальный элемент второй половины = %d\n",
         fmax(mas+k, n2));
  // можно и так вызвать функцию
  // p = mas + k;
  // max2 = fmax(p, n2);
  return 0;
}
```

Задача. *Разработать функцию для нахождения индекса минимального элемента и суммы элементов одномерного массива целых чисел.*

```
int func(int *m, int n, int *sum) {
  int i, min, n_min;
  min = *m;
  n_min = 0;          // индекс минимального равен 0
  *sum = m[0];
  for (i = 1; i < n; i++) {
    if (m[i] < min) {
      min = *(m+i);
      n_min = i;      // находим индекс минимального
    }
    *sum += m[i];     // находим сумму элементов
  }
  return n_min;       // возвращаем индекс минимального элемента
}
```

```
int main() {
  int mas[100], n, i, sum, nmin;
  printf("Введите размерность массива\n");
  scanf("%d", &n);
  printf("Введите массив\n");
  for (i = 0; i < n; i++)
    scanf("%d", &mas[i]);
  nmin = func(mas, n, &sum);
  printf("Номер минимального = %d\n", nmin);
  printf("Сумма = %d\n", sum);
  return 0;
}
```

<u>Задача.</u> Разработать функцию для замены всех нулевых элементов массива заданным числом.

```
void f(int *m, int n, int k) {
  int i;
  for (i = 0; i < n; i++)
    if (m[i] == 0)
      *(m+i) = k;          // m[i] = k;
}

int main() {
  int mas[100], n, i;
  printf("Введите размерность массива\n");
  scanf("%d", &n);
  printf("Введите массив\n");
  for (i = 0; i < n; i++)
    scanf("%d", &mas[i]);
  f(mas, n, 100);          // заменить все 0 на 100
  printf("Массив после замены\n");
  for (i = 0; i < n; i++)
    printf("%d ", mas[i]);
  printf("\n");
  return 0;
}
```

ЛАБОРАТОРНАЯ РАБОТА № 8

Задача. Начиная с текущего каталога, вывести на экран все каталоги и файлы.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <io.h>
#include <time.h>
```

// печать файлов

```
void PrintFile() {
  struct _finddata_t sFile;
  long hFile;
  int pr;
  hFile = _findfirst("*.*", &sFile);
  if (hFile != -1) {
    do {
      if (!(sFile.attrib & _A_SUBDIR))
        printf("                %s\n", sFile.name);
      pr = _findnext(hFile, &sFile);
    }
    while (pr == 0);
    _findclose(hFile);
  }
}
```

// печать каталогов

```
void PrintDir() {
  char pp[256]; //каталог
  struct _finddata_t dir;
  long hDir;
  int pr;
  getcwd (pp, 255);
  printf("%s\n", pp);
  PrintFile();
  hDir = _findfirst("*.*", &dir);
  if (hDir != -1) {
    do {
      if (strcmp(dir.name, ".") != 0 &&
          strcmp(dir.name, "..") != 0 &&
          (dir.attrib & _A_SUBDIR) == _A_SUBDIR) {
        chdir(dir.name);  // заходим в каталог
        PrintDir();
        chdir("..");      // возвращаемся в родительский каталог
      }
      pr = _findnext(hDir, &dir);
    }
    while (pr == 0);
    _findclose(hDir);
```

```
  }
}
int main() {
  char path_beg[256];  // начальный каталог
  char path_end[256];  // конечный каталог
  getcwd (path_beg, 255);      // сохраняем каталог запуска (начальный)
  PrintDir();
  getcwd (path_end, 255);      // получаем конечный каталог
  if (strcmp(path_beg, path_end))
    puts("Ошибка: не вернулись в начальный каталог!");
  puts("Все каталоги и файлы напечатаны!");
  system("pause");
  return 0;
}
```

Задача. Начиная с текущего каталога, удалить все файлы с расширением .bak.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <io.h>
#include <time.h>
```

```
// удаление файлов с расширением .bak
void UnlinkBak() {
  char pp[256]; //каталог
  struct _finddata_t sFile;
  long hFile;
  int pr;
  getcwd (pp, 255);
  printf("%s\n", pp);
  hFile = _findfirst("*.bak", &sFile);
  if (hFile != -1) {
    do {
      if (!(sFile.attrib & _A_SUBDIR)) {
        printf("                %s\n", sFile.name);
        // unlink(sFile.name);    // пока идет отладка, не стоит удалять файлы
      }
      pr = _findnext(hFile, &sFile);
    }
    while (pr == 0);
    _findclose(hFile);
  }
}
```

```
// переход в другой каталог
void NextDir() {
  struct _finddata_t dir;
  long hDir;
  int pr;
```

```
  UnlinkBak();
  hDir = _findfirst("*.*", &dir);
  if (hDir != -1) {
    do {
      if (strcmp(dir.name, ".") != 0 &&
          strcmp(dir.name, "..") != 0 &&
          (dir.attrib & _A_SUBDIR) == _A_SUBDIR) {
        chdir(dir.name);   // заходим в каталог
        NextDir();
        chdir("..");       // возвращаемся в родительский каталог
      }
      pr = _findnext(hDir, &dir);
    }
    while (pr == 0);
    _findclose(hDir);
  }
}
int main() {
  char path_beg[256];  // начальный каталог
  char path_end[256];  // конечный каталог
  getcwd (path_beg, 255);      // сохраняем каталог запуска (начальный)
  NextDir();
  getcwd (path_end, 255);      // получаем конечный каталог
  if (strcmp(path_beg, path_end))
    puts("Ошибка: не вернулись в начальный каталог!");
  puts("Все файлы с расширением .bak удалены!");
  system("pause");
  return 0;
}
```

ЛАБОРАТОРНАЯ РАБОТА № 9

Задача. Кодирование и декодирование строки.

```
#include <stdio.h>
int main() {
  unsigned char s[50] = "0123"; //пример строки
  unsigned char ss[50], sss[50];
  unsigned char c, c1, c2;
  int nbait, nbit, nbit1, i, n;
  // Кодирование строки s
  // добавлением пятого бита 1 к четырём битам каждого полубайта
  // Для примера: исходная строка s = "0123"
  // 0x30   0x31   0x32   0x33
  // 00110000 00110001 00110010 00110011
  // Закодированная строка ss
  // 0x38   0x4E   0x33   0x94   0xE7
  // 00111000 01001110 00110011 10010100 11100111
  nbait = 0;
  nbit = 0;
  memset(ss, 0, 50);
  n = strlen(s);
  for (i = 0; i < n; i++) {
    c1 = s[i] & 0xF0 | 0x08;
    c = c1 >> nbit;
    ss[nbait] |= c;
    nbit += 5;
    if (nbit > 7) {
      nbait++;
      c = c1 << (13 - nbit);
      ss[nbait] = c;
      nbit -= 8;
    }
    c2 = (s[i] << 4) | 0x08;
    c = c2 >> nbit;
    ss[nbait] |= c;
    nbit += 5;
    if (nbit > 7) {
      nbait++;
      c = c2 << (13 - nbit);
      ss[nbait] = c;
      nbit -= 8;
    }
  }
  // Декодирование строки ss
  // удалением пятого бита у каждого полубайта
  nbit1 = 0;
  nbit = 0;
  nbait = 0;
```

```
  memset(sss, 0, 50);
  i = 0;
  while (nbait < n) {
    if (nbit1 < 4) {
      c1 = ss[i] << nbit1;
      c1 &= 0xF0;
      nbit1 += 5;
      if (nbit1 == 8) {
        i++;
        nbit1 = 0;
      }
    }
    else {
      c1 = ss[i] << nbit1;
      c2 = ss[i+1] >> (8 - nbit1);
      c1 &= 0xF0;
      c2 &= 0xF0;
      c1 |= c2;
      i++;
      nbit1 -= 3;
    }
    sss[nbait] |= (c1 >> nbit);
    nbit += 4;
    if (nbit == 8) {
      nbit = 0;
      nbait++;
    }
  }
  printf("\nИсходная строка      : %s", s);
  printf("\nЗакодированная строка: %s", ss);
  printf("\nДекодированная строка: %s", sss);
  system("pause");
  return 0;
}
```

Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный университет
имени Франциска Скорины»

**В. А. КОРОТКЕВИЧ, Л. И. КОРОТКЕВИЧ,
Г. И. БОЛЬШАКОВА**

ПРОГРАММИРОВАНИЕ: ЯЗЫК ПРОГРАММИРОВАНИЯ С

Практическое руководство

для студентов специальности
1-31 03 03-01 «Прикладная математика
(научно-производственная деятельность)»

Гомель
ГГУ им. Ф. Скорины
2019

УДК 004.42(075.8)
ББК 32.973.26-018я73
К 687

Рецензенты:
кандидат технических наук Е. А. Левчук;
кандидат физико-математических наук Е. А. Ружицкая

Рекомендовано к изданию учебно-методическим советом
учреждения образования «Гомельский государственный
университет имени Франциска Скорины»

Короткевич, В.А.

К 687 Программирование: язык программирования С : практическое руководство / В. А. Короткевич, Л. И. Короткевич, Г. И. Большакова; М-во образования Республики Беларусь, Гомельский гос. ун-т им. Ф. Скорины. – Гомель: ГГУ им. Ф. Скорины, 2019. – 45 с.
ISBN 978-985-439-748-1

В настоящем руководстве представлены материалы для практического выполнения лабораторных работ по разделу «Язык программирования С» курса «Программирование». Каждая работа включает в себя условие лабораторной работы с индивидуальными вариантами и необходимые для ее выполнения краткие теоретические сведения с иллюстративными примерами.

Предназначено студентам специальности 1-31 03 03-01 «Прикладная математика (научно-производственная деятельность)» при выполнении лабораторных работ.

УДК 004.42(075.8)
ББК 32.973.26-018я73

ISBN 978-985-439-748-1

Содержание

Введение

Дисциплина «Программирование» ориентирована на подготовку специалиста, умеющего проектировать эффективные алгоритмы решения поставленной задачи, выбирать наиболее подходящие структуры данных, программные и технические средства его реализации и с учетом операционного окружения разрабатывать программные приложения, отвечающие современным требованиям и новейшим компьютерным технологиям.

Сформированные компетенции в области программирования являются базовыми при изучении всех дисциплин специализации, при выполнении курсовых и дипломных работ.

Целью практического руководства по изучению раздела «Язык программирования С» дисциплины «Программирование» является оказание помощи студентам в усвоении учебного материала по программированию на языке С.

В практическое руководство включены теоретические сведения по языку программирования С, а также задания по лабораторным работам и требования к ним.

Выполнение лабораторных работ включает:

– постановку задачи в соответствии с темой лабораторной работы;

– усвоение студентами необходимого теоретического материала по теме лабораторной работы;

– построение алгоритма решения задачи;

– написание программы и ее отладку на компьютере;

– защиту лабораторной работы у преподавателя.

Практическое руководство адресовано студентам специальности 1-31 03 03-01 «Прикладная математика (научно-производственная деятельность)».

1 Лабораторная работа №1. Операторы управления вычислительным процессом

Краткие теоретические сведения

Типы данных. Данные языка С – это переменные и константы.

Основные типы данных:

– целые типы: char, int, short, long;

– вещественные типы: float, double, long double.

Ключевое слово unsigned (беззнаковый) указывает, что целая переменная не может принимать отрицательные значения. Числовым значением переменной типа char является код соответствующего символа.

Основные типы констант:

– строковая: `"123"`, `"Введите число"`;

– символьная: `'a'`, `'+'`;

– целая: `100`, `0x9A`;

– вещественная: `1.`, `2.0`, `7.5`, `.25`.

Ввод-вывод данных. В языке С для ввода-вывода данных используются библиотечные функции, описанные в файле <stdio.h>.

Для вывода информации на экран используется функция printf(). Функция просматривает форматную строку и выводит каждый символ так, как он есть, пока не встретит спецификацию преобразования (указание функции типа выводимой переменной и формата ее вывода). Спецификация преобразования начинается со знака процента (см. таблицу 1).

```
printf("Введите число");
printf("Вы ввели два числа: %d и %d", a, b);
```

В функции printf() на экран можно выводить не только переменные, но и константы, выражения, результаты вызова функций.

```
printf("\n%d %d %d %d", a, 55, (a+100)*2, func());
```

Для ввода информации с клавиатуры используется функция scanf(). Эта функция получает в качестве параметра форматную строку, содержащую одну или несколько спецификаций преобразования, указывающих формат и тип данных, которые должны быть введены. Дополнительные параметры должны быть адресами переменных, в которых введенные данные будут сохраняться. Адрес переменной можно получить с помощью операции взятия адреса & (например, &x).

```
int a;
unsigned long b;
```

```
char c;
scanf("%d %lu", &a, &b);
scanf("%c", &c);
```

Таблица 1 – Спецификации преобразования функций printf() и scanf()

Спецификация	Вид выводимой информации	Тип выводимых данных
%d	выводится целое число	int
%u	выводится целое число	unsigned int
%ld	выводится целое число	long
%lu	выводится целое число	unsigned long
%f	выводится вещественное число	float
%lf	выводится вещественное число	double
%c	выводится символ	char
%s	выводится строка	строка

Операторы ветвления if и else. Позволяют в зависимости от условия выполнить какой-то один фрагмент кода из нескольких.

Оператор if осуществляет условное ветвление программы, проверяя истинность выражения.

```
if (a > b) max = a;
if (a != 100 && b < a) {
  x = a - b;
  y = a + b;
}
```

При необходимости в комбинации с if можно использовать ключевое слово else, позволяющее выполнить альтернативный оператор, если выражение в условии ложно.

```
if (a > b) max = a;
else max = b;
if (a > 0 && b > 0) {
  x = a - b;
  y = a + b;
}
else y = a * b;
```

Операторы if и else могут быть вложенными.

Оператор цикла while. Позволяет выполнять тело цикла до тех пор, пока условие цикла не перестанет быть истинным.

```
while (условие)
  тело цикла;
```

Обычно инициализирующее выражение расположено до цикла, а выражение, которое влияет на условие цикла, находится в конце тела цикла.

```
int i = 0;            // инициализация счётчика цикла
while (i < 10) {
  // тело цикла
  i++;                // изменение счетчика цикла
}
```

Оператор цикла do...while. В этом цикле проверка условия производится после исполнения тела цикла.

```
do
  тело цикла;
while (условие);
int a;
// зацикливание до тех пор, пока не будет введено положительное число
do {
  printf("Введите положительное число\n");
  scanf("%d", &a);
}
while (a <= 0);
```

Оператор цикла for. Это наиболее сложная форма оператора цикла.

```
for (выражение_1; условие; выражение_2)
  тело цикла;
```

Цикл for похож на цикл while в следующей интерпретации:

```
выражение_1;
while (условие) {
  тело цикла;
  выражение_2;
}
```

Любое выражение их двух в заголовке цикла for может быть последовательностью простых операторов, разделяемых оператором запятая. Условие и два выражения в круглых скобках цикла for являются необязательными.

Если внутри условного оператора и операторов цикла находится несколько операторов, то их обязательно надо брать в фигурные скобки.

Оператор break прекращает выполнение цикла, в котором он непосредственно находится.

Лабораторная работа

Задание 1. Разработать программу для решения поставленной задачи, используя условные операторы.

Требования и ограничения:

– задание выполнить, не вводя массивы;

– при необходимости обеспечить контроль введенных данных.

Варианты:

1. Найти сумму наибольшего и наименьшего из трех чисел.
2. Ввести три числа и вывести их на экран в отсортированном по возрастанию порядке.
3. Если для трех чисел сумма двух первых больше произведения двух последних, найти квадрат произведения всех трех чисел, в противном случае – найти утроенное второе число.
4. Определить расстояние от точки x до отрезка [a;b] на прямой. Если точка принадлежит отрезку, считать, что расстояние равно 0.
5. Для трех чисел найти наименьшее целое число, которое больше всех этих чисел.
6. Пусть на прямой заданы два отрезка [a;b] и [c;d]. Найти длину их пересечения.
7. Найти максимальное отрицательное число среди трех чисел. Если среди чисел нет отрицательных, то выдать соответствующее сообщение.
8. Определить, можно ли построить треугольник с заданными длинами сторон.
9. Для двух отрезков [a;b] и [x;y] на прямой определить их взаимное расположение: отрезки не пересекаются, отрезки пересекаются, один отрезок находится внутри другого.
10. Для трех чисел найти номер первого нулевого из них. Если нулевого числа нет, то выдать соответствующее сообщение.
11. Определить, можно ли из четырех длин сторон, введенных в порядке возрастания, построить прямоугольник.
12. Пусть на прямой заданы точка k и отрезок [x;y]. Определить расстояние от точки до ближайшего конца отрезка.
13. Для трех чисел x, y и z определить, сколько из них не попадает в интервал [a;b].
14. Найти наибольшее целое отрицательное число, которое меньше всех трех введенных чисел.
15. Найти минимальное число из четырех чисел.

Задание 2. Разработать программу для обработки последовательности целых чисел, используя оператор цикла while.

Требования и ограничения:

– задание выполнить, не вводя массивы;

– число, служащее признаком окончания ввода, не является членом последовательности.

Варианты:

1. Определить положительный минимум из членов последовательности. Окончание ввода – число 0.
2. Определить разность между количеством четных и нечетных чисел последовательности, не принадлежащих интервалу [-5; 5]. Окончание ввода – число 1000.
3. Определить, сколько раз в последовательности чисел идут подряд два нулевых числа. Окончание ввода – отрицательное число.
4. Найти сумму двузначных чисел последовательности, кратных 3. Окончание ввода – число 0.
5. Найти общее количество элементов последовательности, расположенных между первым и вторым элементами, равными m. Окончание ввода – число 0.
6. Определить наибольшее количество подряд идущих однозначных членов последовательности. Окончание ввода – число 500.
7. Найти минимальное число среди чисел последовательности, больших 10. Окончание ввода – число 210.
8. Найти число последовательности, наиболее далекое по модулю от предшествующего. Окончание ввода – отрицательное число.
9. Определить количество чисел последовательности, делителем которых является ее первый член. Окончание ввода – отрицательное число.
10. Найти сумму чисел последовательности с нечетными номерами, предшествующих первому элементу, равному k. Окончание ввода – число 0 или число, меньшее 100.
11. Найти количество чисел последовательности, находящихся за последним значением, равным n. Окончание ввода – число 0.
12. Найти максимальную разность между соседними членами последовательности. Окончание ввода – число 0.
13. Подсчитать количество четных членов последовательности, превышающих предыдущий член последовательности не менее чем на 5. Окончание ввода – отрицательное число кратное 3.
14. Определить наименьшее число среди чисел последовательности, кратных 4. Окончание ввода – количество однозначных членов последовательности равно 5.
15. Определить, является ли последовательность чисел арифметической прогрессией. Окончание ввода – число, кратное 10.

Задание 3. Используя цикл for, вывести таблицу значений функции f(x) с параметром u с точностью до четырех знаков после точки в n равноудаленных и равномерно распределенных точках на отрезке [a; b].

Требования и ограничения:

– обеспечить контроль корректности данных;

– обеспечить вывод номера ветки (от 1 до 3), по которой вычисляется значение функции.

Варианты:

1. $$f(x)=\begin{cases}\ln(x/(u-5\cdot x)),|x|<10\\ \sqrt{1/u}\cdot u/x, x>30\\ x^4\cdot\sin(x),\text{ в остальных случаях}\end{cases}$$

2. $$f(x)=\begin{cases}(x^5-u)/\ln(x), x\in[1;20]\\ x\cdot\cos(x),|x|>30\\ 3+\sqrt[4]{x+u},\text{ в остальных случаях}\end{cases}$$

3. $$f(x)=\begin{cases}\ln(u\cdot x^2-4), 2<x<10\\ (x+u)/(x+25)^2,|x|>20\\ e^{-x}/x,\text{ в остальных случаях}\end{cases}$$

4. $$f(x)=\begin{cases}\sqrt{u-x^2},|x|<3\\ e^x\cdot x^6, x\in[-10;-5]\\ \sin(\ln(|x|)),\text{ в остальных случаях}\end{cases}$$

5. $$f(x)=\begin{cases}\sqrt{u\cdot x}, -5<x<20\\ \cos^{-2}(\pi\cdot(x+u)),|x|>30\\ x^3+x^2-15,\text{ в остальных случаях}\end{cases}$$

6. $$f(x)=\begin{cases}u/(x-3)^2, -2<x<4\\ \sqrt{x+u/(x-10)^3}, x\in[8;20]\\ \sin(e^x),\text{ в остальных случаях}\end{cases}$$

7. $$f(x)=\begin{cases}\sqrt{u\cdot x},|x|<4\\ \sin(x+u), x>30\\ e^{-x}/(u+x^2),\text{ в остальных случаях}\end{cases}$$

8. $$f(x)=\begin{cases}\sin(x)\cdot(x+e), x\le 1\\ 3^x+\sqrt[3]{x/2}-1/u, x>10\\ 1/\ln(u\cdot x-5),\text{ в остальных случаях}\end{cases}$$

9. $$f(x)=\begin{cases}\sqrt{u\cdot x}\cdot 2^{x+1}\cdot\cos(x), x\le 0\\ -x/\sqrt{|x-5\cdot u|}, 0<x<4\\ e^x\cdot\ln(x^2-25),\text{ в остальных случаях}\end{cases}$$

10. $$f(x)=\begin{cases}\sqrt[3]{x^2}, x\le 0\\ \ln(x-u)\cdot|5-x|, x\ge 5\\ \sin^2(x)/(x-1),\text{ в остальных случаях}\end{cases}$$

11. $$f(x)=\begin{cases}\sin(x+5)/\cos(\pi\cdot(x-1)), x\le 0\\ \sqrt[5]{x^2/5}+\ln(u\cdot x)/20, x>10\\ -x^2\cdot e^{x/3\cdot u},\text{ в остальных случаях}\end{cases}$$

12. $f(x)=\begin{cases} x^3/(u+x), x\in[1;20] \\ e^{|x|/300}, |x|>30 \\ \sqrt[4]{\ln(x)+u}, \text{в остальных случаях} \end{cases}$

13. $f(x)=\begin{cases} x+\cos(x)/(u^2+2), 0<x<50 \\ \ln(2\cdot x+u), x<-2 \\ 2^x/(x+1)^4, \text{в остальных случаях} \end{cases}$

14. $f(x)=\begin{cases} \sqrt{x^3}/u^2, x<5 \\ \ln(3-u\cdot x)^{-1}, x\in[5;10] \\ \sin(e^x), \text{в остальных случаях} \end{cases}$

15. $f(x)=\begin{cases} \sqrt{u\cdot x^2-2}, -2<x<10 \\ \cos(x), x<-7 \\ \ln(u+x)/e, \text{в остальных случаях} \end{cases}$

2 Лабораторная работа №2. Массивы

Краткие теоретические сведения

Массив – это совокупность элементов данных одного типа, объединенных общим именем и расположенных в непрерывной области памяти вплотную друг за другом.

При описании одномерного массива (вектора) за именем массива в квадратных скобках указывается число элементов массива.

```
int m[5], mas[2*40]; //
```
в массиве m – 5 элементов, mas – 80

Имя массива с квадратными скобками, в которых записано целое число, обозначает соответствующий элемент массива. В языке С нумерация элементов массива начинается с нуля.

При описании массив можно инициализировать:

```
int a[5] = {1,2,3,4,5}; //
```
a[0]=1, a[1]=2, a[2]=3, a[3]=4, a[4]=5

В языке С нет встроенных средств для изменения всего массива целиком. Изменяют массив поэлементно с помощью циклов.

```
int mas[10];
for (i = 0; i < 10; i++) // цикл обнуления всего массива
  mas[i] = 0;
```

Нельзя один массив напрямую присвоить другому.

```
int a[10], b[10];
for (i = 0; i < 10; i++) //
```
цикл присвоения массивов

```
  a[i] = b[i];
```

Массив вводят и выводят поэлементно с помощью циклов.

```
int a[10], n = 7;
printf("Введите элементы массива\n");
for (i = 0; i < n; i++)
  scanf("%d", &a[i]);
```

Описание двумерного массива (матрицы):

```
int m[4][3];
```

Имя двумерного массива с двумя выражениями в квадратных скобках за ним обозначает соответствующий элемент двумерного массива.

Инициализация двумерного массива при описании:

```
int a[4][3] = {{ 18, 21,  5 },
               {  6,  7, 11 },
               { 30, 52, 34 },
               { 24,  4, 67 }};
```

Ввод двумерного массива осуществляется поэлементно с помощью двух вложенных циклов.

```
int a[5][10], n = 5, m = 10;
for (i = 0; i < n; i++)
  for (j = 0; j < m; j++)
    scanf ("%d", &a[i][j]);
```

Вывод двумерного массива также осуществляется с помощью двух вложенных циклов.

```
for (i = 0; i < n; i++) {
  for (j = 0; j < m; j++)
    printf ("%5d ", a[i][j]);
  printf("\n"); //  переход на следующую строку
}
```

В двумерном массиве строка, столбец и диагонали в квадратной матрице являются по своей сути одномерными массивами. Поэтому при их обработке достаточно использовать один цикл, а не два вложенных.

Лабораторная работа

Задание 1. Разработать программу для обработки массива.

Требования и ограничения:

– исходные данные имеют тип int;

– обеспечить контроль корректности ввода размерности исходных массивов (при неверном вводе пользователь должен иметь возможность вводить размерность до верного ввода).

Варианты:

1. Дан массив A[n]. Найти среднее арифметическое элементов массива, кратных заданному числу k.
2. Дан массив A[n]. Определить, сколько раз в массиве встречается минимальный элемент.
3. Дан массив A[n]. Найти номер такого элемента массива, который имеет максимальную сумму предыдущего и последующего элементов.
4. Дан массив A[n]. Преобразовать массив таким образом, чтобы сначала располагались все элементы, имевшие нечетные индексы, а затем элементы с четными индексами.
5. Дан массив A[n]. Определить номера первого положительного и последнего отрицательного элементов массива.
6. Дан массив A[n]. Найти наибольшее количество одинаковых идущих подряд элементов массива.
7. Дан массив A[n]. Найти количество элементов массива между первым по порядку нулевым и последним по порядку нулевым элементами массива.
8. Дан массив A[n]. Сжать массив, удалив из него все элементы, кратные заданному числу k. Освободившиеся в конце массива элементы заполнить нулями.
9. Дан массив A[n]. Переместить все нулевые элементы в начало массива, не меняя порядок следования других элементов.
10. Дан массив A[n]. Найти сумму элементов массива без двух наибольших элементов.
11. Дан массив A[n]. Подсчитать количество элементов массива с четными индексами, больших по абсолютной величине, чем k.
12. Дан массив A[n]. Найти сумму нечетных элементов, расположенных между первым и последним положительными элементами.
13. Дан массив A[n]. Найти произведение абсолютных величин нечетных элементов массива с четными индексами.
14. Дан массив A[n]. Найти количество элементов, расположенных до первого элемента кратного 5 и после последнего элемента кратного 5.
15. Дан массив A[n]. Найти сумму модулей элементов, расположенных между первым и вторым нулевыми элементами массива.

Задание 2. Разработать программу для обработки массива.

Требования и ограничения:

– исходные данные имеют тип int;

– обеспечить контроль корректности ввода размерности исходных массивов (при неверном вводе пользователь должен иметь возможность вводить размерность до верного ввода);

– реализовать вывод результатов до и после сортировки.

Варианты:

1. Дана матрица A[n][m]. Сформировать вектор B из элементов строк матрицы, в которых содержится максимальный элемент матрицы. Отсортировать вектор B по убыванию.

2. Дана матрица A[n][m]. Среди столбцов матрицы, содержащих только такие элементы, которые по модулю не больше 10, найти столбец с минимальным произведением элементов. Отсортировать найденный столбец матрицы по возрастанию.

3. Дана матрица A[n][n]. Расположить на главной диагонали для каждого столбца сумму элементов. Сформировать вектор B из строки, в которой элемент на главной диагонали имеет максимальной значение. Отсортировать вектор B по убыванию.

4. Дана матрица A[m][n]. Поэлементно вычесть заданную строку матрицы из тех строк матрицы, в которых нет элемента с заданным значением t. Отсортировать по убыванию строку полученной матрицы, в которой больше всего нулевых элементов.

5. Дана матрица A[n][m]. Поменять местами две заданные строки матрицы, предварительно переставив элементы в обратном порядке в той из них, сумма элементов в которой минимальна. Отсортировать первый столбец полученной матрицы по возрастанию.

6. Дана матрица A[n][n]. Сформировать вектор B из максимальных элементов таких столбцов матрицы, в которых нет отрицательных элементов. Отсортировать вектор B по убыванию.

7. Дана матрица A[n][n]. Сформировать вектор B из элементов матрицы, значения которых по модулю больше среднего арифметического элементов матрицы из столбцов, в которых нет нулевых элементов. Отсортировать вектор B по возрастанию.

8. Дана матрица A[n][m]. В строках матрицы, содержащих только элементы из заданного диапазона [a;b], каждый элемент, кроме первого, заменить суммой всех предыдущих. Отсортировать заданный столбец полученной матрицы по убыванию.

9. Дана матрица A[n][n]. Найти все номера строк матрицы, для которых произведение нулевого и диагонального элементов максимально. Удалить из матрицы первую строку. Отсортировать заданный столбец полученной матрицы по убыванию.

10. Дана матрица A[n][n]. Заменить элементы побочной диагонали суммами модулей элементов соответствующих столбцов. Помнять местами первую и последнюю строки матрицы. Отсортировать главную диагональ полученной матрицы по возрастанию.

11. Дана матрица A[n][n]. Получить вектор B путем поэлементного умножения строки и столбца матрицы, где находится её максимальный элемент. Отсортировать вектор B по убыванию.

12. Дана матрица A[n][n]. Определить, совпадают ли в матрице главная и побочная диагонали. Получить вектор B из элементов строки и столбца матрицы, где находится ее максимальный элемент. Отсортировать вектор B по возрастанию.

13. Дана матрица A[n][m]. Найти в матрице столбец с минимальной суммой положительных элементов. Сформировать вектор B из этого столбца. Отсортировать вектор B по возрастанию.

14. Дана матрица A[m][n]. Удалить из матрицы последнюю строку, в которой содержится максимальный элемент матрицы, сформировав из этой строки вектор B. Отсортировать вектор B по возрастанию.

15. Дана матрица A[n][n]. Поэлементно вычесть столбец минимального элемента матрицы из всех строк матрицы. Сформировать вектор B из нулевых элементов матрицы. Отсортировать главную диагональ полученной матрицы по убыванию.

3 Лабораторная работа №3. Указатели

Краткие теоретические сведения

Указатели. При описании любой переменной в памяти выделяется непрерывная область для хранения этой переменной. Номер байта, с которого начинается в памяти переменная, называется адресом переменной.

Указатель – переменная, значением которой является адрес в памяти. Указатель может содержать адреса данных только определенного типа.

Примеры описания указателей:

```
int *pa, *pb;
double *p;
char *s;
```

Есть две специальные операции для работы с указателями.

Унарная операция & выдает адрес объекта. Операция & применяется только к объектам, расположенным в памяти.

```
int x, *px;
px = &x;
```

Оператор присваивает указателю px адрес переменной x (говорят, что теперь px указывает на x).

Унарная операция * называется операцией разыменования. Результатом операции * является значение того объекта, к адресу которого применялась эта операция, а тип результата совпадает с типом объекта.

Операция * может применяться только к указателям.

```
int *p, a, b;
int **pp;
p = &a;
*p = 10;    // a = 10
p = &b;
*p = 20;    // b = 20
pp = &p;
**pp = 30; // b = 30
```

Указателю нельзя присвоить число. Но значение 0 (NULL) может быть присвоено указателям любого типа. Считается, что указатель, равный нулю, ни на что не указывает.

Адресная арифметика. Для указателей-переменных разрешены следующие операции: присваивание, сравнение, сложение и вычитание, инкремент ++ и декремент --.

Указателю можно присвоить адрес переменной или указатель того же типа. Любому указателю можно присвоить значение NULL.

```
int *a, *b;
double *d;
a = b;              // правильно
d = a;              // ошибка
```

Язык С разрешает операцию сравнения указателей, при этом сравниваются адреса в памяти.

Сравнение указателей с помощью операций <, <=, >, >= в общем случае некорректно. Но, если указатели указывают на элементы одного и того же массива, то такие операции работают правильно.

Сравнение указателей с помощью операций == и != в любом случае работает правильно. Любой указатель можно сравнить на равенство или неравенство с NULL.

Над указателями определены операции сложения и вычитания с целым числом. Если к указателю, описанному как type *p;, прибавляется или отнимается константа N, его значение изменяется на N*sizeof(type).

```
int *p, i = 2;
int a[5] = {1,2,3,4,5};
p = &a[1];  // *p = 2
p++;        // *p = 3, значение p увеличится на 2
```

```
p += i;       //  *p = 5, значение p увеличится на 4
p--;          //  *p = 4, значение p уменьшится на 2
p += 2;       //  *p = 2, значение p уменьшится на 4
```

Два указателя на элементы одного и того же массива можно вычитать. Разность двух указателей type *p1, *p2; – это разность их значений, поделенная на sizeof(type). Результат может быть отрицательным.

Указатели и массивы. Любую операцию, которую можно выполнить с помощью индексов массива, можно сделать и с помощью указателей.

Имя массива – это указатель-константа на начало массива.

```
int *p;
int mas[10];
```

Установка указателя p на начало массива mas:

```
p = mas;
p = &mas[0];
```

Если указатель-константа mas указывает на нулевой элемент массива, то mas+1 указывает на следующий, а mas+i – на i-й элемент массива.

```
mas   == &mas[0]
mas+1 == &mas[1]
mas+i == &mas[i]
```

Если mas+i это адрес mas[i], то *(mas+i) есть содержимое mas[i].

```
*mas     == mas[0]
*(mas+1) == mas[1]
*(mas+i) == mas[i]
```

Если указатель p является указателем на элемент массива, то в выражениях его также можно использовать с индексом: p[i] идентично *(p+i).

```
int a[10] = {0,1,2,3,4,5,6,7,8,9};
int *pa, i = 5, x;
pa = a;
x = *a;           //  x = 0, т.е. x = a[0]
x = *(a+i);       //  x = 5, т.е. x = a[i] i=5
*(a+9) = 99;      //  a[9] = 99
x = *pa;          //  x = 0, т.е. x = a[0]
x = *(pa+3);      //  x = 3, т.е. x = a[3]
*(pa+7) = *(a+2); //  a[7] = a[2];
int *p = a+2;     //  устанавливаем p на a[2]
x = p[3];         //  x = 5 или x = a[5]
p[0] = *(p+7);    //  a[2] = a[9];
```

Выражение a[i] всегда преобразуется к виду *(a+i), а доступ к элементу одномерного массива физически реализуется через адрес начала массива в памяти и смещение элемента (индекс элемента) от начала массива.

Доступ к элементам двумерного массива также может осуществляться как по индексам, так и с помощью указателей.

Выражение m[i][j] всегда преобразуется к виду: *(адрес начала массива + смещение), где смещение определяется уже двумя индексами i и j.

Имя двумерного массива с одним индексным выражением является указателем-константой на соответствующую строку массива.

Для двумерного массива int m[2][3]; указателями-константами на нулевую и первую строки будут m[0] и m[1]:

```
m[0] == &m[0][0]
m[1] == &m[1][0]
```

Для доступа к элементам двумерного массива используются два индексных выражения:

```
m[1][2] = *(m[1]+2) = *(*(m+1)+2) = (*(m+1))[2]
```

Сместимся от начала массива на одну строку, взяв адрес этой строки из указателя-константы, а затем сместимся уже в нужной строке на нужное количество элементов.

Лабораторная работа

Задание 1. Выполнить задание 1 из лабораторной работы №2, используя для доступа к элементам массива указатели.

Требования и ограничения:

– в программе операцию [] использовать нельзя.

Задание 2. Выполнить задание 2 из лабораторной работы №2, используя для доступа к элементам массивов указатели.

Требования и ограничения:

– в программе операции [] и [][] использовать нельзя.

4 Лабораторная работа №4. Строки

Краткие теоретические сведения

В языке С нет отдельного типа для строк. Работа со строками реализована через массивы. Символьная строка – это одномерный массив типа char, заканчивающийся символом '\0'.

При объявлении строки в виде char s[10]; для хранения строки будет выделено 10 байт памяти. В этой строке s можно хранить строки длиной от 0 до 9 символов, поскольку один символ требуется для хранения символа '\0', завершающего строку.

К каждому символу строки можно обращаться как к элементу массива: s[0], s[1] и т.д.

Строковая константа – это набор символов в двойных кавычках.

Длина строки – это число символов, предшествующих символу '\0'.

Пустая строка – это строка с нулевой длиной. Чтобы получить пустую строку, надо в нулевой символ строки занести символ '\0'.

Ввод и вывод строк с помощью функций scanf() и printf():

```
char s[80];
scanf("%s", s);
printf("%s", s);
```

Есть специальные функции для ввода и вывода только строк:

```
gets(s);
puts(s);
```

При выводе на экран строки выводятся все символы строки вплоть до завершающего нулевого символа. Нулевой символ на экран не выводится.

При вводе строка автоматически дополняется символом '\0'.

При описании строки можно инициализировать.

```
char s[80] = "университет";
```

Поскольку имя массива является указателем на первый элемент массива, переменные типа строк можно считать, как имеющие тип char *. Но это не значит, что, написав char *s, мы описываем строку. Мы описываем указатель на строку. В этот указатель можно занести адрес реальной строки (размещенной в памяти) или строковой константы.

```
char s[80];   //  описали строку
char *ps;     //  описали указатель на строку
ps = s;       //  указатель ps стал указывать на строку s
gets(ps);     //  введенная строка сохранится в памяти,
              //  выделенной для переменной s
char *p;      //  описали указатель и в него занесли
p = "12345";  //  адрес константы "12345"
              //  Изменять такую строку-константу нельзя!!!
```

При выполнении операции присваивания в ячейки памяти, отведенные для указателя, пересылается не массив символов, а только указатель на его начала, т.е. адрес первого символа строковой константы.

```
s = p;        //  ошибка, т.к. s – указатель-константа
```

```
ps = p;      //  в ps занесли адрес константы "12345"
```

Массив строк – обычный двумерный массив символов.

Описание массива строк:

```
char mas[5][10]; // будет выделена память для хранения 5 строк
```

Имя массива с одним индексом является строкой символов.

Массив строк можно инициализировать при описании:

```
char mas [5][10] = {"май", "июль", "август"};
```

Можно обрабатывать строки посимвольно, обращаясь к отдельным элементам строки через их индексы.

Как определить, что символ является цифрой:

```
if (s[i] >= '0' && s[i] <= '9') { ... }  // цифра
```

В языке С нет встроенных операций для работы со строками, но есть библиотечные функции для обработки строк. Прототипы функций для обработки строк описаны в файле <string.h>.

Все функции для строк в качестве параметров используют адреса строк, а не сами строки. В функцию передается адрес, который она считает адресом начала строки, и обрабатывает строку, начиная с этого адреса и до символа '\0'.

Определение длины строки: `int strlen(char *s);`

Возвращается количество символов в строке до символа '\0'.

Слияние строк: `char *strcat(char *s1, char *s2);`

К строке, на которую указывает s1, приписываются все символы строки s2, включая и символ '\0'. Функция возвращает адрес строки s1.

Копирование строк: `char *strcpy(char *s1, char *s2);`

Строка, на которую указывает s2, включая символ '\0', копируется в строку, на которую указывает s1. Функция возвращает адрес строки s1.

Сравнение строк: `int strcmp(char *s1, char *s2);`

Функция возвращает значение больше нуля, если строка s1 больше s2 в алфавитном порядке, меньше нуля, если строка s1 меньше s2, и равное нулю, если строки равны.

Занесение символа: `char *strset(char *s, char ch);`

Помещает символ ch во все позиции строки s. Возвращает указатель на строку s.

Поиск подстроки: `char *strstr(char *s1, char *s2);`

Отыскивает место первого вхождения строки s2 в строку s1. Возвращает указатель на начало вхождения. Если строка не найдена, возвращает NULL.

Поиск символа: `char *strchr(char *s, char ch);`

Функция осуществляет поиск символа ch с начала строки, на которую указывает s, и возвращает адрес найденного символа. Если символ не найден, возвращает NULL.

Реверсирование строки: `char *strrev(char *s);`

Функция реверсирует строку, на начало которой указывает s, и возвращает указатель на полученную строку.

Примеры вызова функций:

```
char s1[120], s2[120], s3[120], *p;
int len;
gets(s1);
len = strlen(s1);
strcpy(s2, s1);
gets(s3);
strcat(s2, s3);
if ((p = strchr(s3, 'b')) != NULL) {...}
if ((p = strstr(s3, "def")) == NULL) {...}
if (!strcmp(s1, "abc")) {...}
if (strcmp(s1+1, s2) <=0 ) {...}
```

Лабораторная работа

Задание 1. Разработать программу для посимвольной обработки символьной информации.

Требования и ограничения:

– реализовать программу, не используя функции для обработки символьной информации.

Варианты:

1. Дано слово. Добавить в начало слова столько последних символов слова, сколько всего есть таких символов в слове.
2. Дано слово. Вставить между одинаковыми символами символ '=', а между разными – символ '*'.
3. Дано слово. Заменить в слове все маленькие латинские буквы кроме 'z' на следующие по алфавиту.
4. Дано слово. Удалить из слова все нечетные цифры.
5. Дано слово. Повторяющиеся подряд символы удалить, оставив лишь один из них.
6. Дано слово. Определить, состоит ли слово из различных неповторяющихся символов.
7. Дано слово. Поменять символы слова попарно местами.
8. Дано слово. Определить, состоит ли слово из одних цифр.

9. Дано слово. Заменить в слове все большие латинские буквы на соответствующие маленькие латинские буквы.

10. Дано слово. Вставить в слове после каждой цифры символ '!'.

11. Дано слово. После каждой группы повторяющихся символов вставить символ '_'.

12. Дано слово. Если первый символ слова совпадает с последним символом, то удалить из слова два первых и два последних символа.

13. Дано слово. Удвоить буквы на нечетных позициях в слове.

14. Дано слово. Записать все символы слова в обратном порядке и добавить в начало и конец слова символ '?'.

15. Дано слово. Если первый символ слова цифра, то вставить ее после каждого знака '+' в слове.

Задание 2. Разработать программу для обработки символьной информации с помощью функций для работы со строками.

Требования и ограничения:

– посимвольное обращение к элементам строки допустимо только для проверки на '\0' и для установки '\0';

– не требовать ввода количества слов в массиве (признаком окончания ввода массива является пустая строка).

Варианты:

1. Дан массив слов. Преобразовать исходный массив, вставив в каждое слово длиной более двух символов после второй буквы подстроку из двух начальных букв этого же слова. Сформировать предложение из тех слов полученного массива, которые являются «перевертышами».

2. Дан массив слов и подстрока. Сформировать предложение из слов, содержащих заданную подстроку не более двух раз. В предложение должны войти слова только четной длины.

3. Дан массив слов, каждое из которых начинается с трех цифр. Сформировать предложение из слов, длина которых более шести символов, и последние четыре символа которых не содержат подстроку "**", упорядочив в предложении слова по возрастанию первых трех символов.

4. Дан массив слов и одно слово. Сформировать предложение из слов, которые содержат в своем составе после третьего символа перевернутое исходное слово, упорядочив слова в предложении по убыванию количества символов в слове.

5. Дан массив слов и предложение. Сформировать новое предложение из слов массива, которые не входят в заданное предложение, причем во всех словах, длина которых строго больше четырех, удалить по два символа в начале и в конце слова.

6. Дан массив слов. Сформировать предложение из слов, в составе каждого из которых дважды встречается или подстрока "****", или подстрока "++", причем встречающиеся подстроки не пересекаются.

7. Дан массив слов. Преобразовать исходный массив слов, укоротив слова с начала слова на количество символов в предыдущем слове, если длина предыдущего слова меньше. Сформировать предложение из слов, длина которых четная.

8. Дан массив слов. Сформировать предложение из слов, длина которых больше длины предыдущего слова, но меньше длины последующего, сместив их символы по кругу на три позиции вправо.

9. Дан массив слов и подстрока. Преобразовать исходный массив слов, удалив из слов все группы цифр длиной больше двух. Сформировать предложение из слов, которые заканчиваются заданной подстрокой.

10. Дан массив слов и слово. Сформировать предложение из таких слов массива, которые имеют в своем составе только символы из заданного слова и длина которых больше двух символов. Слова в предложении должны быть отсортированы по первым трем символам по алфавиту.

11. Дан массив слов и подстрока. Сформировать новое слово из двух начальных букв тех слов, где подстрока встречается по меньшей мере два раза после третьей буквы.

12. Дан массив слов и подстрока. Каждое четное слово преобразовать, удвоив его. Сформировать предложение из слов, которые содержат заданную подстроку после k-го символа.

13. Дан массив слов. Сформировать предложения из слов, в которых первые k букв совпадают. В результате – массив предложений.

14. Дан массив слов. Сформировать предложение из слов, которые не являются перевертышами и не состоят из одних цифр.

15. Дан массив слов и слово. Сформировать предложение из слов, входящих в заданное слово не более одного раза.

5 Лабораторная работа №5. Функции

Краткие теоретические сведения

Функции. Функция – это последовательность операторов, которые определены и записаны только в одном месте программы, однако их можно вызвать для выполнения из одной или нескольких точек программы.

```
//  функция для вычисления суммы двух целых чисел
int sum(int a, int b) {
```

```
  int s = a + b;
  return s;
}
```

Сначала указан тип значения, которое функция возвращает, – int. Затем следует имя функции – sum. После имени функции в круглых скобках перечислены формальные параметры функции с указанием их типов. Операторы в фигурных скобках называются телом функции.

Возвращение значения из функции. Перед именем функции задается тип возвращаемого функцией значения. Если тип возвращаемого значения задан как void, считается, что функция не возвращает никакого значения.

Возврат в вызвавшую функцию с передачей значения из вызванной функции происходит с помощью оператора возврата return.

```
// функция для нахождения минимума из двух целых чисел
int min(int a, int b) {
  int m = a;
  if (b < a) m = b;
  return m;
}
```

Операторов return в функции может быть несколько.

```
// функция для нахождения минимума из двух целых чисел
int min(int a, int b) {
  if (a < b) return a;
  return b;
}
```

В операторе return можно записывать выражения.

```
// функция для вычисления суммы двух целых чисел
int sum(int a, int b) {
  return a + b;
}
```

После слова return можно ничего не записывать. В этом случае вызвавшей функции никакого значения не передается (тип функции – void).

```
void f() {
  ...
  return;   // можно опустить
}
```

В функции, тип которой отличен от void, в операторе return необходимо обязательно указать возвращаемое значение. И завершение такой функции обязательно должно быть по оператору return.

Формальные и фактические параметры функции. После имени функции в круглых скобках указываются формальные параметры функции с указанием их типов. В теле функции формальными параметрами пользуются так же, как обычными переменными.

```
// функция для вычисления суммы двух целых чисел
int sum(int a, int b) {
  a += b;
  return a;
}
```

Функция может быть и без параметров.

```
// функция для вывода на экран сообщения об ошибке
void prnErr() {
  printf("Ошибка!");
}
```

Параметры, передаваемые функции при ее вызове, называются фактическими. При вызове функции фактическими параметрами могут быть: имена переменных, выражения или просто константы. Значения фактических параметров заносятся в соответствующие формальные параметры.

Вызов функции. Когда имя функции встречается в операторе программы, говорят, что в этой точке функция вызывается. Вызов функции выполняет ее тело. А затем возвращается управление в точку, находящуюся непосредственно за вызовом функции.

Если функция не имеет тип void, ее вызов может использоваться как операнд в выражении.

```
int main() {
  int x = 10, y = 7, res;
  res = sum(x, y);
  // вызов функции в выражениях
  res = sum(x, y) + sum(5, 9) + sum(x+y, 100);
  if (sum(x, y) > 0)
    printf("Сумма положительная");
  if ((res = sum(x, y)) < 0)
    printf("Сумма отрицательная = %d", res);
  printf("%d", sum(x, y));   // печатаем возврат
  sum(x, y);                 // игнорируем возврат
  return 0;
}
```

Объявление и определение функции, прототип функции. Определение функции – это ее полное описание и ее тело. Объявление функции (прото-

тип функции) содержит лишь ту информацию, которая необходима компилятору, чтобы избегать ошибок при вызове функции.

```
//  прототип функции для вычисления суммы двух целых чисел
int sum(int a, int b);
```

Имена параметров можно не указывать.

```
//  прототип функции для вычисления суммы двух целых чисел
int sum(int, int);
```

Прототип функции не нужен, если определение функции находится в том же файле, что и вызов функции, но до первого вызова этой функции.

Обычно прототипы функций собирают в заголовочный файл, а потом этот файл при необходимости включают в нужные файлы с помощью директивы #include.

```
//  подключение файла с прототипами системных функций
#include <stdio.h>
//  подключение файла с прототипами собственных функций
#include "my.h"
```

Передача параметров в функцию main(). Функция main() может иметь параметры. Информация передается функции main() с помощью параметров командной строки. Параметры командной строки – это информация, которая вводится в командной строке вслед за именем программы при запуске программы на выполнение. Обычно параметры командной строки используют для того, чтобы передать программе начальные данные, которые понадобятся ей при запуске.

При передаче параметров в функцию main() ее надо определять так:

```
int main(int argc, char *argv[]) {
  ...
  return 0;
}
```

Параметр argc содержит количество параметров в командной строке.

Параметр argv является массивом указателей на строки. В этом массиве каждый элемент указывает на очередной параметр командной строки. Первый параметр – имя программы с полным путем.

```
// вывести на экран все параметры командной строки
int main(int argc, char *argv[]){
  for (int i = 0; i < argc; i++)
    printf("%s\n", argv[i]);
  return 0;
}
```

Все параметры командной строки являются строковыми, поэтому преобразование числовых параметров в нужный формат должно быть предусмотрено в программе при ее разработке.

```
// запуск программы с двумя параметрами: task.exe 100 2.7
int main(int a, char *b[]) {
  int k;
  double f;
  // проверка, что все параметры командной строки заданы
  if (argc != 3) {
    printf("Неверное количество параметров!\n");
    return 0;
  }
  k = atoi(b[1]);
  f = atof(b[2]);
  ...
  return 0;
}
```

Передача параметров в функции в языке С. В языке С есть только один способ передачи параметров в функции – передача по значению.

Метод передачи по значению реализуется следующим способом:

– формальный параметр рассматривается как локальная переменная функции, т.е. память для него выделяется в стеке при входе в функцию;

– вызывающая функция вычисляет фактический параметр и помещает его значение в память, выделенную для формального параметра.

Функция может использовать, а также изменять значения своих формальных параметров. Но при выходе из функции формальные параметры перестают существовать. Функция не может изменить значения переменных, указанных как фактические параметры при вызове данной функции, т.к. она не имеет доступа к этим переменным.

Передача указателей в функции. Если функция должна изменять значение фактического параметра, то в функцию надо передавать не значение этого параметра, а его адрес. При передаче адреса параметр функции следует описывать как один из типов указателей. Функция может записать все, что угодно туда, куда направлен переданный указатель.

```
// функция для перестановки местами значений двух переменных
void swap(int *pa, int *pb) { // параметры-указатели
  int temp;
  temp = *pa;    // сохранить значение a
  *pa = *pb;     // занести b в a
```

```
  *pb = temp;    // занести a в b
}
int main() {
  int i = 10, j = 20;
  int *pi = &i, *pj = &j;
  printf("i и j перед обменом: %d %d\n", i, j);
  swap(&i, &j);// передаем адреса переменных i и j
  printf("i и j после обмена: %d %d\n", i, j);
  swap(pi, pj);// передаем адреса переменных i и j через указатели
  printf("i и j после обмена: %d %d\n", i, j);
  return 0;
}
```

Передача одномерных массивов в функции. При указании имени массива в качестве параметра при вызове функции в функцию передается не сам массив, а адрес нулевого элемента массива.

```
int main() {
  int mas[10];
  ...
  func(mas);  // в функцию передается адрес начала массива
  ...
  return 0;
}
```

Т.к. функции передается адрес начала массива, то функция работает с реальным содержимым этого массива и может изменить это содержимое.

Для обработки массива, кроме адреса начала массива функция должна знать и количество элементов в массиве. Поэтому в функцию дополнительно следует передавать точное число элементов в массиве.

```
// функция для замены всех 0 в массиве заданным числом
void f(int *m, int n, int k) {
  for (int i = 0; i < n; i++)
    if (m[i] == 0) *(m+i) = k;  // m[i] = k;
}
int main() {
  int mas[100], n, i;
  printf("Введите размерность массива\n");
  scanf("%d", &n);
  printf("Введите массив\n");
  for (i = 0; i < n; i++) scanf("%d", &mas[i]);
  f(mas, n, 100);  // заменить все 0 на 100
  printf("Массив после замены\n");
```

```
  for (i = 0; i < n; i++) printf("%d ", mas[i]);
  return 0;
}
```

Передача в функции символьных строк. Передача символьной строки в функцию подобна передаче одномерного массива. Реально в функцию передается адрес строки. Но, в отличие от обычного одномерного массива, для строк нет необходимости передавать в функцию их реальный размер, так как все строки оканчиваются нулевым символом.

```
// функция для вычисления длины строки
int lenStr(char *s) {
  int n;
  for (n = 0; s[n]; n++);
  return n;
}
int main() {
  char str[80];
  gets(str);
  printf("Длина строки = %d\n", lenStr(str));
  return 0;
}
```

```
// функция для получения адреса первого вхождения символа в строку
char *func(char c, char *s) {
  for (; *s; s++)
    if (c == *s) return s;
  return NULL;
}
int main() {
  char ss[80], *p;
  gets(ss);
  p = func('a', ss);
  if (p != NULL) printf("Символ найден: %s\n", p);
  else printf("Символ не найден\n");
  return 0;
}
```

Возвращение указателей из функций. Есть несколько тонкостей при разработке функций, которые возвращают указатели:

– нельзя объявлять возвращаемый тип как int *, если возвращается указатель типа char *;

– надо очень внимательно следить за тем, чтобы объект, адрес которого возвращает функция, существовал после выхода из нее.

Есть три подхода к решению проблемы существования объекта, адрес которого возвращает функция:

– делать такие объекты глобальными;

– описывать такой объект в вызывающей функции и передавать его в вызываемую функцию;

– описывать такие объекты в функции как static.

```
//  функция для удаления в строке заданного символа
char *delChar(char *s, char c) {
  char ss[80];
  int i = 0, j = 0;
  while (s[i]) {
    if (s[i] != c) ss[j++] = s[i];
    i++;
  }
  ss[j] = 0;
  for (i = 0; i <= j; i++) s[i] = ss[i];
  return s;  // return ss; – ошибка
}
int main() {
  char str[80];
  gets(str);
  puts(delChar(str,'a'));
  return 0;
}
//  функция для сцепления двух строк
char *strcat(char *s, char *ss) {
  static char str[500];  // обязательно надо static
  int i = 0, j = 0;
  while(s[i]) str[j++] = s[i++];
  for (i = 0; ss[i]; i++) str[j++] = ss[i];
  str[j] = 0;
  return str;
}
int main() {
  char s1[80], s2[80], *p;
  gets(s1); gets(s2);
  p = strcat(s1, s2);  // адрес можно сохранить
  puts(p);
  return 0;
}
```

Основные сведения о файлах. Файл – это линейная последовательность байтов, объединенная одним именем.

Файлы бывают текстовые и двоичные. Текстовый файл – это файл, в котором каждый символ хранится в виде одного байта (кода, соответствующего символу). Двоичный файл – файл, данные которого представлены в двоичном виде.

Возможны два режима доступа к файлу: текстовый и двоичный. В текстовом режиме производится преобразование пары символов 0x0D 0x0A – при чтении из файла в один символ новой строки '\n', а при записи в файл обратно в пару символов. В двоичном режиме никакого преобразования байтов не происходит.

Доступ к файлу может быть последовательным и прямым. Последовательный доступ – файл читается/пишется последовательно байт за байтом. Прямой доступ – чтение/запись возможны из произвольного места файла с установкой указателя чтения/записи в нужное место файла.

Открытие и закрытие файлов. Функция fopen() открывает файл name в заданном режиме mode.

```
FILE *fopen(char *name, char *mode);
```

Функция fopen() возвращает указатель файла – указатель на структуру FILE. Эта структура содержит всю необходимую информацию о файле (его имя, открыт ли файл для чтения или записи, указатель текущей позиции в файле, не встретился ли конец файла). Если при открытии файла происходит ошибка, то fopen() возвращает пустой указатель (NULL).

Режимы открытия файла:

– открыть файл для чтения: r;
– открыть файл для записи: w, a;
– открыть файл для чтения/записи: r+, w+, a+.

Для указания на текстовый режим работы с файлом после любого из этих значений может быть добавлен символ 't', для указания на двоичный режим – символ 'b'.

```
FILE *fp;
if ((fp = fopen("test.txt", "w")) == NULL) {
  printf("Ошибка открытия файла!\n");
  return 0;  // завершение работы программы
}
```

Функция fclose() закрывает файл, который был открыт с помощью функции fopen().

```
int fclose(FILE *fp);
```

где fp – указатель файла, возвращенный в результате вызова fopen().

При нормальном завершении работы программы для каждого открытого файла fclose() вызывается автоматически

Форматированный ввод/вывод. Форматированный ввод/вывод выполняется с помощью функций fscanf() и fprintf().

```
int fprintf(FILE *fp, char *format, ...);
int fscanf(FILE *fp, char *format, ...);
```

Задача. Вывести в файл, а затем ввести из файла 5 строк. Каждая строка содержит данные разных типов.

```
int main() {
  FILE *f;
  int i, k;
  double d;
  char s[] = "Line";
  if ((f = fopen("file.txt", "w")) == NULL) {
    printf("Ошибка открытия файла\n");
    return 0;
  }
  for (i = 1; i <= 5; i++) {
    fprintf(f, "%s %d %.2lf\n", s, i, sqrt(i));
    printf("%s %d %.2lf\n", s, i, sqrt(i));
  }
  fclose(f);
  if ((f = fopen("file.txt ", "r")) == NULL) {
    printf("Ошибка открытия файла\n");
    return 0;
  }
  for (i = 1; i <= 5; i++) {
    fscanf(f, "%s %d %lf", s, &k, &d);
    printf("%s %d %.2lf\n", s, k, d);
  }
  fclose(f);
  return 0;
}
```

Другие средства для работы с файлами. Функция feof() позволяет определить, достигнут ли конец файла.

```
int feof(FILE *fp);
```

Функция возвращает ненулевое значение, если при операции ввода из файла был обнаружен конец файла, и 0 в противном случае.

Задача. Ввести из файла неопределенное количество чисел типа long.

```
int main(void) {
  FILE *fp;
  long t;
  if ((fp = fopen("in", "r")) == NULL) {
    printf("Ошибка открытия файла\n");
    return 0;
  }
  fscanf(fp, "%ld", &t);
  while (!feof(fp)) {
    printf("%ld\n", t);
    fscanf(fp, "%ld", &t);
  }
  fclose(fp);
  return 0;
}
```

Лабораторная работа

Задание 1. Разработать две функции для вычисления заданной величины и вызвать их из функции main().

Требования и ограничения:

– исходные данные должны вводиться в функции main();

– первая функция должна возвращать заданную величину;

– во второй функции обеспечить контроль правильности исходных данных (кроме вычисления заданной величины, функция должна возвращать признак правильности исходных данных);

– в функции main() вывести результат работы первой функции и результат работы второй функции или сообщение об ошибке.

Варианты:

1. Координата середины отрезка $[x_1;x_2]$ на прямой.
2. Сумма n первых членов арифметической прогрессии.
3. Площадь трапеции по основаниям и высоте.
4. Площадь боковой поверхности цилиндра по радиусу и высоте.
5. Объем прямоугольного параллелепипеда по трем сторонам.
6. Площадь круга по диаметру.
7. Площадь ромба по его диагоналям.
8. Объем пирамиды по площади основания и высоте.
9. Длина диагонали прямоугольного параллелепипеда по сторонам.
10. Длина окружности по радиусу.
11. Длина средней линии трапеции по основаниям.
12. Площадь полной поверхности цилиндра по радиусу и высоте.
13. Расстояние между двумя несовпадающими точками на плоскости.

14. Площадь поверхности шара по его диаметру.
15. Сумма внутренних углов выпуклого n-угольника.

Задание 2. Разработать функцию для обработки одномерного массива для задания 1 из лабораторной работы №2 и вызвать ее из функции main().

Требования и ограничения:

– ввод размерности массива и самого массива должен осуществляться из текстового файла в функции main();

– результаты работы программы функции main() вывести в текстовый файл произвольного формата;

– имена исходного и результирующего файлов должны вводиться с клавиатуры.

Задание 3. Разработать функцию для обработки символьной информации для задания 1 из лабораторной работы №4 и вызвать ее из функции main().

Требования и ограничения:

– слова для обработки передаются в программу как параметры командной строки;

– если это необходимо по условию задачи, все остальные исходные данные должны вводиться с клавиатуры;

– разработанная функция должна вызываться столько раз, сколько параметров передано в функцию main();

– результаты работы программы в функции main() вывести в текстовый файл произвольного формата.

6 Лабораторная работа №6. Работа с динамической памятью

Краткие теоретические сведения

Понятие динамического объекта. Данные, которые создаются и уничтожаются по требованию программиста, называются динамическими. Динамические объекты не имеют имен, и работа с ними выполняется только с помощью указателей, в которых хранятся их адреса. Память под динамические объекты выделяется в куче (heap). Количество динамических объектов в программе ограничивается только размером кучи.

Создание и уничтожение динамических объектов. Прототипы функций для работы с динамической памятью содержатся в файле <stdlib.h>.

Функции malloc() и calloc() служат для динамического выделения памяти, т.е. для создания динамических объектов. Эти функции возвращают адрес созданного динамического объекта.

Выделить память объемом size байт:

```
void *malloc(unsigned long size);
```

Возвращается указатель на выделенную память или NULL в случае неудачи. Выделенная память содержит случайные значения.

Выделить память под n элементов по size байт каждый:

```
void *calloc(unsigned long n, unsigned long size);
```

Возвращается указатель на выделенную память или NULL в случае неудачи. Выделенная память обнуляется.

```
char *ps = (char *)malloc(100);
if (ps == NULL) return 0;
int *pi = (int *)calloc(1, sizeof(int));
if (pi == NULL) return 0;
```

Функция realloc() служит для изменения размера ранее выделенного блока памяти, адрес которого содержится в указателе block. Параметр size задает новый размер блока.

```
void *realloc(void *block, unsigned long size);
```

Функция возвращает указатель на перезахваченный блок памяти.

```
// захватывает блок памяти для 50 символов
char *ps = (char *)malloc(50);
if (ps == NULL) return 0;
// перезахватывает блок для 100 символов
ps = (char *)realloc(ps, 100);
if (ps == NULL) return 0;
```

Для явного освобождения памяти используется функция free().

```
void free(void *block);
```

Функция освобождает область памяти, на которую указывает block. В указателе должен содержаться адрес области памяти, полученной с помощью malloc(), calloc() или realloc(). Нельзя повторно освобождать те области памяти, которые уже освобождены.

После освобождения считается, что память больше не принадлежит программе. Поэтому ее использование является ошибкой.

```
fpee(p);
pp = p->next;  // ошибка
```

Объект, созданный при помощи функций динамического выделения памяти, существует или до тех пор, пока он не удален явно при помощи

функции освобождения памяти, или до конца работы программы.

К динамическим объектам нельзя обратиться по имени, так как они просто не имеют имени. У нас есть только указатель с адресом динамического объекта. Поэтому для доступа к самому динамическому объекту используется операция разыменования *:

```
int *pi = (int *)calloc(1, sizeof(int));
if (pi == NULL) return 0;
*pi = 55;
```

Динамическое выделение памяти для одномерных массивов. Одномерный массив из n целых чисел можно создать двумя способами:

```
int *a, *b, n = 10;
a = (int *)malloc(n * sizeof(int));
b = (int *)calloc(10, sizeof(int)); // инициализация 0
```

Освобождение памяти, выделенной под массивы:

```
free(a); free(b);
```

Адреса освобождаемых массивов должны строго совпадать с адресами, которые вернули функции динамического размещения массивов.

```
int *a, n = 10;
a = (int *)malloc(n * sizeof(int));
a += 2;     // перешли к элементу массива с индексом 2
*a = 100;   // a[2] = 100
free(a);    // ошибка, т.к. a != адрес начала массива
```

Пример программы для работы с динамическим одномерным массивом целых чисел.

```
// функция для выделение памяти для массива и ввода массива
int vvod(int **m) {
  int i, nn;
  scanf ("%d", &nn);  // ввод размерности массива
  // выделение памяти под массив нужного размера
  *m = (int *)calloc(nn, sizeof(int));
  if (*m == NULL) return 0;
  // ввод массива
  for (i = 0; i < nn; i++) scanf("%d", *m+i);
  return nn;  // возвращаем размерность созданного массива
}
// поиск минимального элемента в массиве
int fmin(int *m, int nn) {
  int i, min = 0x7FFFFFFF;
```

```
  for (i = 0; i < nn; i++)
    if (m[i] < min) min = *(m+i);
  return min;
}
int main() {
  int n, i, min, *b;   // указатель для массива b[n]
  n = vvod(&b);   // передается адрес указателя, а не его значение
  if (n == 0) return 0;
  // печать массива
  for (i = 0; i < n; i++) printf("%d ", b[i]);   // b[i]
== *(b+i)
  min = fmin(b, n);
  printf("\nmin = %d\n", min);
  free(b);   // освобождение памяти
  return 0;
}
```

Лабораторная работа

Задание. Выполнить задание 1 из лабораторной работы №2, размещая все массивы в программе только динамически.

Требования и ограничения:

– ввод размерности массива и самого массива осуществлять в функции main() из текстового файла с именем, заданным в командной строке;

– для основного алгоритма обработки массива, а также для ввода и вывода массива разработать функции.

7 Лабораторная работа №7. Пользовательские типы данных

Краткие теоретические сведения

Основные сведения о структурах. Структуры – это одна или несколько переменных, обычно описывающих какую-нибудь сущность, которые для удобства работы с ними сгруппированы под одним именем.

Шаблон структуры определяет новый пользовательский тип данных:

```
struct имя_шаблона_структуры {
  описание_элементов
};
```

Перечисленные в структуре переменные называются полями структуры. Имена полей в одном шаблоне должны быть уникальными.

```
//  шаблон структуры «Студент» (ФИО, год рождения, адрес)
struct Student {
  char fio[80];
  int year;
  char address[100];
};
//  шаблон структуры «Точка на плоскости» (пара целых координат)
struct Point {
  int x;
  int y;
};
```

На основе шаблона можно создать много структурных переменных.

```
struct Student stud = {"Иванов", 1990, "Гомель"};
struct Student stud1;
struct Point pnt1, pnt2 = {5, 7};
```

В языке С реализован ограниченный набор операций над структурами как единым целым: передача структуры в качестве параметра функции, возврат структуры из функции, получение адреса структуры, определение указателя на структуру.

Структуры нельзя сравнивать.

Можно присваивать одну структуру другой, если они соответствуют одному шаблону.

```
MYPOINT pnt1, pnt2 = {5, 5};
pnt1 = pnt2;
```

Доступ к полям структуры. Для доступа к отдельным полям структурной переменной используется операция '.':

```
имя_структурной_переменной.имя_поля
```

Можно описывать указатели на структуры. Для доступа к полям структуры через указатель используется операция '->' вместо операции '.'.

Ссылка на поле структурной переменной может располагаться в любом месте выражения, точно так же, как и простая переменная.

```
struct Point pnt1, pnt2 = {5, 7};
pnt1.x = 1; pnt1.y = 10;
printf("Точка 2: x=%d y=%d", pnt2.x, pnt2.y);
float dist;  // расстояние между точками 1 и 2
dist = sqrt((pnt1.x - pnt2.x) * (pnt1.x - pnt2.x)+
            (pnt1.y - pnt2.y) * (pnt1.y - pnt2.y));
```

```
struct Point *ptr = &pnt2;
ptr->x = 0;   // (*ptr).x = 0;  pnt2.x = 0;  (&pnt2)->x = 0;
```

Массивы структур. Можно описывать массивы структур:

```
struct Student grp[35];
```

Массив структур можно инициализировать при описании:

```
struct Student grp1[3]= {{"Иванов", 1990, "Гомель"},
                         {"Петров", 1989, "Минск"},
                         {"Зайцев", 1990, "Гомель"}};
```

Доступ к элементам массива структур:

```
strcpy(grp[i].fio, "Иванов");
(*(grp+i)).year = 1990;
strcpy((grp+i)->address, "Гомель");
```

Для доступа к элементам массива структур может использовать и обычный указатель:

```
struct Student *ptr = grp;
strcpy(ptr[i].fio, "Иванов");
(*(ptr+i)).year = 1990;
strcpy((ptr+i)->address, "Гомель");
ptr++;  // переход к студенту с номером i+1
```

```
// Описать группу студентов, заполнить информацию о группе и
// студентах группы с клавиатуры, определить самого юного студента в
// группе, отсортировать студентов группы по фамилии.
int main() {
  int i, j;
  struct Student {
    char fio[80];
    int year;
    char adr[100];
  };
  struct Gruppa {
    char name[10];   // название группы
    int num;         // количество студентов в группе
    struct Student stud[35];  // студенты группы
  } grp;
  struct Student *min, st;
  printf("Название группы: "); scanf("%s", grp.name);
  printf("Кол-во студентов: ");scanf("%d", &grp.num);
  printf("Введите информацию о студентах: \n");
  for (i = 0; i < grp.num; i++) {
```

```
    printf("Студент %d: \n", i);
    printf("Фамилия: "); gets(grp.stud[i].fio);
    printf("Год рождения: ");
    scanf("%d", &grp.stud[i].year);
    printf("Адрес: "); gets(grp.stud[i].adr);
  }
  // поиск самого юного студента
  if (grp.num == 0) min = NULL;
  else min = &grp.stud[0];
  for (i = 1; i < grp.num; i++)
    if (grp.stud[i].year >  min->year)
      min = &grp.stud[i];
  if (min) {
    printf("Самый юный студент: \n");
    printf("Фамилия = %s\n", min->fio);
    printf("Год рождения = %d\n", min->year);
    printf("Адрес = %s\n", min->adr);
  }
  else
    puts("В группе нет студентов!\n");
  // сортировка студентов по фамилии
  for (i = 0; i < grp.num-1; i++)
    for (j = i+1; j < grp.num; j++) {
      char *psi = grp.stud[i].fio;
      char *psj = grp.stud[j].fio;
      if (strcmp(psi, psj) > 0) {
        st = grp.stud[i];
        grp.stud[i] = grp.stud[j];
        grp.stud[j] = st;
      }
    }
  printf("Отсортированный список %s:\n", grp.name);
  for (i = 0; i < grp.num; i++) {
    printf("Фамилия = %s\n", grp.stud[i].fio);
    printf("Год рождения = %d\n", grp.stud[i].year);
    printf("Адрес = %s\n", grp.stud[i].adr);
  }
  return 0;
}
```

Структуры и функции. В функцию можно передавать: поля структуры по отдельности, всю структуру целиком и указатель на структуру.

```
void print1(char *f, int y, char *a) {
  printf("Фамилия = %s\n", f);
  printf("Год рождения = %d\n", y);
  printf("Адрес = %s\n", a);
}
void print2(struct Student s) {
  printf("Фамилия = %s\n", s.fio);
  printf("Год рождения = %d\n", s.year);
  printf("Адрес = %s\n", s.adr);
}
void print3(struct Student *p) {
  printf("Фамилия = %s\n", p->fio);
  printf("Год рождения = %d\n", p->year);
  printf("Адрес = %s\n", p->adr);
}
struct Student st = {"Иванов", 1990, "Гомель"};
print1(st.fio, st.year, st.adr);
print2(st);
print3(&st);
```

Если функция должна изменять структурную переменную, в нее следует передавать указатель на модифицируемую структуру. Без ограничений можно возвращать из функции структуру и указатель на структуру.

```
// Пример программы для работы с динамическим одномерным
// массивом структур: создание массива, ввод, вывод на печать, поиск
// книги с минимальной стоимостью, освобождение памяти.
int main() {
  struct Book {
    char name[20];
    int cost;
  };
  struct Book *p, min;
  p = (struct Book *)malloc(5 * sizeof(struct Book));
  for (int i = 0; i < 5; i++)
    scanf("%s %d", (p+i)->name, &(p+i)->cost);
  for (int i = 0; i < 5; i++)
    printf("%s %d\n", p[i].name, p[i].cost);
  min = p[0];
  for (int i = 1; i < 5; i++)
    if (min.cost > (p+i)->cost) min = *(p+i);
  printf("min: %s %d\n", min.name, min.cost);
  free(p);
```

```
  return 0;
}
```

Лабораторная работа

Задание. Для заданного объекта описать шаблон структуры в соответствии с заданием варианта и выполнить следующие действия:

– описать структурную переменную, заполнить ее с клавиатуры и вывести ее на экран;

– динамически создать структурную переменную, заполнить ее с клавиатуры и вывести ее на экран, удалить созданную переменную;

– описать одномерный массив структурных переменных, заполнить его с клавиатуры, выполнить требуемые действия в соответствии с заданием А варианта, вывести на экран полученные результаты;

– ввести размерность массива, динамически создать одномерный массив структурных переменных, заполнить его с клавиатуры, выполнить требуемые действия в соответствии с заданием Б варианта, вывести на экран полученные результаты, удалить созданный массив.

Требования и ограничения:

– для ввода с клавиатуры и вывода на экран отдельной структурной переменной разработать функции.

Варианты:

1. Объект – книга (название, год издания, тип: детская, научная, художественная, техническая, другое). А – отсортировать книги по возрастанию названия. Б – найти количество книг заданного типа, год издания которых равен максимальному году издания среди всех книг.

2. Объект – школа (номер, количество учащихся, тип: средняя, лицей, гимназия, другое). А – отсортировать школы по убыванию количества учащихся. Б – найти общее количество лицеев и гимназий с количеством учащихся из заданного диапазона.

3. Объект – наблюдение за климатом (дата, температура, вид осадков: нет, дождь, снег, град). А – отсортировать наблюдения по виду осадков. Б – найти среднюю температуру дождливых дней заданного года.

4. Объект – самолет (марка, бортовой номер, примечание: принадлежит аэропорту, взят в аренду, личный, другое). А – отсортировать самолеты по возрастанию названия марки. Б – для взятых в аренду самолетов найти максимальный бортовой номер самолетов заданной марки.

5. Объект – дом (название улицы, номер дома, тип: многоквартирный, частный, другое). А – отсортировать дома по убыванию названия улицы. Б – найти количество различных типов домов, номер которых находится в заданном диапазоне.

6. Объект – собака (возраст, кличка, вид: служебная, декоративная, охотничья, другая). А – отсортировать собак по кличкам. Б – найти количество различных кличек собак заданного возраста.

7. Объект – улица (длина в километрах, название, тип: центральная, магистральная, пешеходная, переулок, проезд). А – отсортировать улицы по убыванию длины. Б – найти общую длину переулков и пешеходных улиц, в названии которых есть цифры.

8. Объект – телепередача (название, рейтинг, жанр: политика, экономика, кино, музыка, юмор, спорт, другое). А – отсортировать телепередачи по возрастанию рейтинга. Б – найти максимальный и минимальный рейтинг юмористических передач.

9. Объект – стипендия студента (фамилия, сумма, вид: учебная, повышенная, именная, социальная). А – отсортировать стипендии по фамилии студента. Б – найти суммарную стипендию студентов, фамилии которых начинаются с заданной буквы и заканчиваются на заданную букву.

10. Объект – товар (название, цена, товарная группа: мясные, молочные, кондитерские, консервы, напитки, другое). А – отсортировать товары по возрастанию цены. Б – найти товар заданной товарной группы, название которого содержит заданную букву и цена которого минимальна.

11. Объект – газета (название, год основания, периодичность выхода: ежедневная, 5 раз в неделю, 1 раз в неделю, 1 раз в месяц). А – отсортировать газеты по убыванию года основания. Б – найти минимальную разность в годах основания двух любых еженедельных газет.

12. Объект – музыкальный альбом (исполнитель, длительность, тип музыки: классическая, для детей, джаз, рок, популярная). А – отсортировать альбомы по типу. Б – найти суммарную длительность альбомов заданного исполнителя с классической и популярной музыкой.

13. Объект – выставка (название, количество экспозиций, тип: универсальная, специализированная, художественная). А – отсортировать выставки по типу. Б – найти общее количество экспозиций на выставках, в названии которых нет цифр.

14. Объект – городской транспорт (номер маршрута, длина маршрута в километрах, вид: метро, автобус, троллейбус, трамвай). А – отсортировать транспорт по возрастанию длины маршрута. Б – найти маршрут заданного вида, длина которого максимальна.

15. Объект – абонент мобильной связи (фамилия, оплата в месяц, подключенные услуги: интернет, роуминг, телевидение, отсутствуют). А – отсортировать абонентов по фамилии. Б – найти минимальную оплату абонентов, имеющих подключенные услуги.

Литература

1 Касаткин, А.И. Профессиональное программирование на языке Си: От Turbo C к Borland C++ / А.И. Касаткин, А.Н. Вальвачев. – Мн: Выш. шк., 1992. – 378 с.

2 Касаткин, А.И. Профессиональное программирование на языке Си: Управление ресурсами / А.И. Касаткин. – Мн: Выш. шк., 1992. – 432 с.

3 Керниган, Б. Язык программирования С / Б. Керниган, Д. Ритчи. – М.: Вильямс, 2017. – 304 с.

4 Шилд, Г. С. Полное руководство / Г. Шилд. – М.: Вильямс, 2017. – 704 с.

5 Прата, С. Язык программирования С. Лекции и упражнения / С. Прата. – М.: Вильямс, 2015. – 928 с.

6 Гриффитс, Д. Изучаем программирование на С / Д. Гриффитс, Д. Гриффитс. – М.: ЭКСМО, 2013. – 624 с.

7 Дейтел, П. Как программировать на С / П. Дейтел, Х. Дейтел. – М.: Бином, 2014. – 1008 с.

8 Шилд, Г. Полный справочник по С / Г. Шилд. – М.: Вильямс, 2009. – 704 с.

9 Полубенцева, М.И. С/С++. Процедурное программирование / М.И. Полубенцева. – СПб.: БХВ-Петербург, 2017. – 432 с.

Производственно-практическое издание

Короткевич **Владимир Апполонович,**
Короткевич **Людмила Ивановна,**
Большакова **Галина Ивановна**

ПРОГРАММИРОВАНИЕ: ЯЗЫК ПРОГРАММИРОВАНИЯ С

Практическое руководство

для студентов специальности
1-31 03 03-01 «Прикладная математика
(научно-производственная деятельность)»

Редактор *В. И. Шкредова*
Корректор *В. В. Калугина*

Подписано в печать 20.10.2019. Формат 60x84 1/16.
Бумага офсетная. Ризография. Усл. печ. л. 2,6.
Уч.-изд. л. 2,8. Тираж 25 экз. Заказ 878.

Издатель и полиграфическое исполнение:
учреждение образования
«Гомельский государственный университет
имени Франциска Скорины»
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 1/87 от 18.11.2013.
Специальное разрешение (лицензия) № 02330 / 450 от 18.12.2013.
Ул. Советская, 104, 246019, Гомель.

3.1. ПЕРЕЧЕНЬ ВОПРОСОВ К ЗАЧЕТУ ПО ДИСЦИПЛИНЕ «ПРОГРАММИРОВАНИЕ» (часть 2)

1. Основные этапы реализации программ на языке С
2. Базовые элементы языка программирования С, комментарии и их использование
3. Целочисленные и вещественные типы данных
4. Элементарный ввод-вывод данных
5. Основные операции языка С
6. Условный оператор if и инструкция множественного выбора switch
7. Операторы цикла в языке С
8. Понятие указателя в языке С, операции взятия адреса и разыменования
9. Определение и инициализация массивов, одномерные и двумерные массивы
10. Методы доступа к элементам массивов
11. Понятие строки в языке С, объявление и инициализация строк
12. Доступ к элементам строки по индексу и через указатели
13. Функции в языке С: имя и тело функции
14. Возврат значений из функций и передача параметров в функции
15. Передача указателей в функции и возврат указателей из функций
16. Прототипы функций
17. Локальные и глобальные переменные, их области видимости
18. Многофайловая компиляция (проекты)
19. Шаблон структуры и структурные переменные, доступ к элементам структуры
20. Статическое и динамическое распределение памяти, функции динамического выделения и освобождения памяти

3.2. ПЕРЕЧЕНЬ ВОПРОСОВ К ЭКЗАМЕНУ ПО ДИСЦИПЛИНЕ «ПРОГРАММИРОВАНИЕ» (часть 2)

1. Основные этапы разработки программ на языке С
2. Базовые элементы языка программирования С, комментарии и их использование
3. Концепция типа данных: встроенные и пользовательские типы данных, простые и структурированные типы данных
4. Целочисленные типы данных, представление в памяти целочисленных типов данных
5. Вещественные типы данных
6. Элементарный ввод-вывод данных
7. Элементарный ввод-вывод данных
8. Преобразование типов при присваивании и в выражениях, явное приведение типов
9. Условный оператор if и инструкция множественного выбора switch
10. Операторы цикла в языке С
11. Бесконечные циклы. Инструкции break и continue, особенности их работы в различных конструкциях циклов
12. Понятие указателя в языке С
13. Операторы взятия адреса и разыменования
14. Арифметические операции с указателями, сравнение указателей
15. Указатели на указатели
16. Определение и инициализация массивов, одномерные массивы
17. Определение и инициализация массивов, одномерные массивы
18. Указатели и одномерные массивы
19. Указатели и двумерные массивы
20. Методы доступа к элементам массивов
21. Указатели на массивы, массивы указателей
22. Понятие строки в языке С, объявление и инициализация строк
23. Доступ к элементам строки по индексу и через указатели
24. Функции в языке С, имя и тело функции
25. Возврат значений из функций
26. Передача параметров в функции
27. Механизмы передачи параметров в вызываемую функцию и из вызываемой функции
28. Передача указателей в функции и возврат указателей из функций
29. Прототипы функций
30. Передача параметров в функцию main()
31. Использование структур в функциях
32. Файлы в языке С, двоичные и текстовые файлы, режимы открытия файлов

33. Библиотека графики
34. Организация рекурсии в языке С, поиск файлов в каталогах
35. Функции для работы с блоками памяти
36. Стандартные процедуры обработки списков
37. Понятие динамической структуры данных
38. Функции преобразования данных
39. Управление файлами, директориями и накопителями
40. Директивы препроцессора условной компиляции
41. Реализация и виды списков
42. Многофайловая компиляция (проекты)
43. Функции для работы со строками символов
44. Доступ к файлам через поток ввода-вывода
45. Статические переменные
46. Префиксный доступ к файлам
47. Стандартные математические функции
48. Препроцессор языка С
49. Директивы препроцессора включения файлов
50. Локальные и глобальные переменные, их области видимости
51. Директивы препроцессора определения макрокоманд (макросов)
52. Динамическое выделение памяти для массивов и структур
53. Статическое и динамическое распределение памяти, функции динамического выделения и освобождения памяти
54. Оператор typedef описания собственного типа данных
55. Доступ к элементам структуры, вложенные структуры
56. Объединения (union)
57. Область определения и видимость идентификатора
58. Перечисления (enum)
59. Массивы структурных переменных
60. Шаблон структуры и структурные переменные

В билете два теоретических вопроса и задача.

3.3. ПРИМЕРЫ ЭКЗАМЕНАЦИОННЫХ ЗАДАЧ ПО ДИСЦИПЛИНЕ «ПРОГРАММИРОВАНИЕ» (часть 2)

Задача 1. Пусть на прямой заданы точка `k` и отрезок `[x;y]`. Определить расстояние от точки до ближайшего конца отрезка.

Задача 2. Определить расстояние от точки `x` до отрезка `[a;b]` на прямой. Если точка принадлежит отрезку, считать, что расстояние равно 0.

Задача 3. Если два отрезка `[a;b]` и `[x;y]` на прямой не пересекаются, то определить, какой из них находится на прямой правее.

Задача 4. Определить, сколько раз в последовательности встречается ее минимальный элемент. Окончание ввода – число из интервала `[-5;-1]`.

Задача 5. Определить, имеются ли в последовательности два идущих подряд нулевых числа. Окончание ввода – отрицательное число.

Задача 6. Найти количество членов последовательности после последнего нулевого элемента. Окончание ввода – отрицательное число.

Задача 7. Удалить из числа цифры кратные 3.

Задача 8. Циклически сдвинуть цифры в записи числа на один разряд влево.

Задача 9. Даны два числа. Добавить к первому числу перевернутое второе число.

Задача 10. Вычислить значение функции `f(x)` в точке x:

$$f(x)=\begin{cases} x^2 - x/5, & \text{если } x > 7 \\ -5 + x^2, & \text{если } x < -1 \\ 2x + 7x - 2, & \text{если } -1 \le x \le 7 \end{cases}$$

Задача 11. Вычислить значение функции `f(x)` в точке x:

$$f(x)=\begin{cases} x/2 - 1, & \text{если } x < 5 \\ -5, & \text{если } x = 5 \\ x + x^2 + 3, & \text{если } x > 5 \end{cases}$$

Задача 12. Вычислить значение функции `f(x)` в точке x:

$$f(x)=\begin{cases} 10 + (x-2)/x, & \text{если } x < -10 \\ 100, & \text{если } x = -10 \\ x/2 * 7 - x^2, & \text{если } x > -10 \end{cases}$$

Задача 13. Дан массив `A[n]`. Найти сумму элементов массива, расположенных между первым и вторым нулевыми элементами массива.

Задача 14. Дан массив `A[n]`. Поменять местами части массива относительно первого отрицательного элемента.

Задача 15. Дан массив `A[n]`. Найти наибольшее количество одинаковых идущих подряд элементов массива.

Задача 16. Дана матрица `A[n][n]`. Добавить в матрицу `A` строку, каждый элемент которой равен количеству нулевых элементов в соответствующем столбце исходной матрицы.

Задача 17. Даны матрица `A[n][m]` и вектор `B[n]`. Определить, является ли вектор `B` столбцом матрицы. Отсортировать заданную строку матрицы по возрастанию.

Задача 18. Дано слово. Если слово не заканчивается на символ '`!`', то вставить в слове перед каждой цифрой символ '`?`'.

Задача 19. Дано слово. После каждой группы повторяющихся символов вставить символ '_'.

Задача 20. Дано слово. Если слово не заканчивается на символ '`!`', то вставить в слове перед каждой цифрой символ '`?`'.

Задача 21. Дано слово. Повторяющиеся подряд символы удалить, оставив лишь один из них.

Задача 22. Дано слово. Удалить из слова все символы '`?`', добавив в конец слова символ '`!`'.

Задача 23. Написать функцию для вычисления заданной величины по значениям параметров функции: площадь трапеции по основаниям и высоте ($S = \frac{a+b}{2}h$). Ввод исходных данных, вызов разработанной функции и вывод результата выполнять в функции `main()`.

Задача 24. Написать функцию для вычисления заданной величины по значениям параметров функции: координата середины отрезка на прямой ($x_{cp} = \frac{x_1 + x_2}{2}$). Ввод исходных данных, вызов разработанной функции и вывод результата выполнять в функции `main()`.

Задача 25. Написать функцию для вычисления заданной величины по значениям параметров функции: расстояние между двумя точками на прямой $r = |x_2 - x_1|$. Ввод исходных данных, вызов разработанной функции и вывод результата выполнять в функции `main()`.

Задача 26. Дан массив `A[n]`. Сформировать массив `B` из положительных элементов массива `A`, а массив `C` – из отрицательных элементов массива `A`. Все массивы размещать динамически. Для доступа к элементам массивов использовать указатели.

Задача 27. Дан массив `A[n]`. Определить, сколько раз в массиве встречается нулевой элемент. Свормировать массив `B` из индексов нулевых элементов. Все массивы размещать динамически. Для доступа к элементам массивов использовать указатели.

Задача 28. Для объекта «*Кот*», состоящего из полей: кличка (`char[]`), порода (`char[]`), стоимость (`float`), описать шаблон структуры и выполнить следующие действия: динамически создать структурную переменную; заполнить ее с клавиатуры и вывести ее на экран; изменить породу и стоимость кота; если кличка кота заканчивается на заданную букву, уменьшить стоимость кота в 2 раза; удалить созданную переменную.

Задача 29. Для объекта «*Спортсмен*», состоящего из полей: ФИО (`char[]`), вес (`int`), рост (`int`), описать шаблон структуры и выполнить следующие действия: динамически создать структурную переменную; заполнить ее с клавиатуры и вывести ее на экран; изменить вес и ФИО спортсмена; если рост спортсмена больше 200 см., то увеличить его вес на 10 кг.; удалить созданную переменную.

Задача 30. Для объекта «*Собака*», состоящего из полей: возраст (`int`), кличка (`char[]`), порода (`char[]`), описать шаблон структуры и выполнить следующие действия: динамически создать структурную переменную; заполнить ее с клавиатуры и вывести ее на экран; изменить кличку и возраст собаки; если название породы собаки начинается с заданной буквы, возраст собаки увеличить на 2 года; удалить созданную переменную.

3.4. ТЕСТЫ И ОБРАЗЕЦ ТЕСТОВЫХ ЗАДАНИЙ ПО ДИСЦИПЛИНЕ «ПРОГРАММИРОВАНИЕ» (часть 2)

По адресу http://dot3.gsu.by/ находятся тесты по дисциплине.

Образец тестовых заданий

1. Язык С. Какое зарезервированное слово указывает, что целая переменная НЕ может принимать отрицательные значения

- o unsigned
- o нет такого зарезервированного слова
- o другое
- o positive
- o long

2. Язык С. Какое значение имеет переменная, которая была определена, но не инициализирована

- o 0 или пустая строка
- o случайное
- o переменная значения не имеет
- o 0x00

3. Язык С. Какой из следующих операторов – оператор сравнения двух переменных на равенство

- o =
- o equal
- o ==
- o :=
- o другое

4. Язык С. С помощью какого оператора можно ввести с клавиатуры значение в переменную, описанную следующим образом:

```
int kol;
```

- o `scanf("%d", &kol);`
- o `scanf(&kol);`
- o `scanf("%d", kol);`
- o `scanf("%f", &kol);`
- o `read(kol);`

5. Язык С. Укажите значение переменной `a` после выполнения следующих операторов:

```
int a, s = 0;
for (a = 0; a < 10; a++) {
  s += 2*a;
}
```

- 9
- 11
- 10
- 1
- значение переменной будет не определено
- 0

6. Язык С. Укажите описания переменных вещественного типа

- `char d;`
- `double e;`
- `long c;`
- `float a;`
- `long double b;`

7. Язык С. Оператор `break`, расположенный в теле цикла, вызывает

- переход к следующей итерации цикла
- выход из обработки исключения
- выход из цикла
- переход к оператору, следующему сразу после текущего цикла

8. Язык С. Какие индексы имеют первый и последний элементы массива

```
int m[20];
```

- 0 и 19
- 1 и 20
- 0 и 20
- 1 и 21

9. Язык С. Укажите тип возвращаемого значения следующей функции

```
int func(char x, float v, double t);
```

- `int`
- `double`
- `char`
- `float`

10. Язык С. Какая функция используется для сравнения двух строк

- `stringcompare()`
- `charcompare()`
- `compare()`
- `cmp()`

- strcmp()

11. Язык C. Укажите строки, в которых происходит обращение к пятому по порядку элементу массива mas:

```
int mas[10];
```

- *mas+4
- mas[5]
- *(mas+4)
- mas+5
- mas(5)
- mas[4]

12. Язык C. Какое значение будет у переменной t после выполнения следующих операторов:

```
int x[5];
int t = sizeof(x) / sizeof(int);
```

- 0
- 20
- 1
- неопределенное
- 5

13. Язык C. Для определения структуры необходимо использовать ключевое слово

- record
- object
- struct
- class
- structure

14. Язык C. Укажите операторы, после выполнения которых значение переменной b уменьшится на 5

- b = b-5;
- b = --b-4;
- b = ++b-5;
- b -= 5;
- B = b-5;
- b = -5+b;

15. Язык C. Укажите правильные операторы динамического выделения памяти для массива из 20 целых чисел

- int *p = calloc(20, sizeof(int));

- `int p = (int *)malloc(20 * sizeof(int));`
- `int *p = (float *)calloc(20, sizeof(int));`
- `float *p = (int *)malloc(20 * sizeof(int));`
- `int *p = (int *)malloc(20 * sizeof(int));`
- `int *p = (int *)calloc(20, sizeof(int));`

4.1. УЧЕБНАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Учреждение образования
«Гомельский государственный университет имени Франциска Скорины»

УТВЕРЖДАЮ
Проректор по учебной работе
ГГУ имени Ф. Скорины
_____ И.В. Семченко
20.05.2020

Регистрационный № УД-12-2020-29 /уч.

ПРОГРАММИРОВАНИЕ

Учебная программа учреждения высшего образования
по учебной дисциплине для специальности
1-31 03 07-01 Прикладная информатика
(программное обеспечение компьютерных систем)

Гомель 2020

Учебная программа составлена на основе образовательного стандарта высшего образования ОСВО 1-31 03 07-2013 и учебного плана учреждения высшего образования по специальности 1-31 03 07-01 «Прикладная информатика (программное обеспечение компьютерных систем)» регистрационный № G 31-01-20/УП от 15.04.2020 г.

СОСТАВИТЕЛИ:

М.С. Долинский – доцент кафедры математических проблем управления и информатики учреждения образования «Гомельский государственный университет имени Франциска Скорины», кандидат технических наук, доцент;

В.А. Короткевич – доцент кафедры математических проблем управления и информатики учреждения образования «Гомельский государственный университет имени Франциска Скорины», кандидат технических наук, доцент;

Л.И. Короткевич – старший преподаватель кафедры математических проблем управления и информатики учреждения образования «Гомельский государственный университет имени Франциска Скорины»

РЕКОМЕНДОВАНА К УТВЕРЖДЕНИЮ:

Кафедрой математических проблем управления и информатики
УО «Гомельский государственный университет имени Франциска Скорины»

(протокол № 10 от 12.05 2020);

Научно-методическим советом УО «Гомельский государственный университет имени Франциска Скорины»

(протокол № 6 от 20.05. 2020)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Учебная программа дисциплины компонента учреждения высшего образования «Программирование» разработана для студентов специальности 1-31 03 07-01 «Прикладная информатика (программное обеспечение компьютерных систем)».

Дисциплина «Программирование» ориентирована на обучение студентов базовым знаниям, умениям и навыкам в области программирования. Изучаемые темы базируются на использовании современных информационных технологий, программного и технического обеспечения компьютеров.

Целью изучения дисциплины является обучение студентов теоретическим основам и практическим навыкам проектирования эффективных алгоритмов решения поставленных задач, разработки на процедурных и объектно-ориентированных языках программирования программных приложений, отвечающих современным требованиям.

Задачами дисциплины являются:

– овладение студентами теоретическими основами программирования;

– овладение студентами методами и приемами разработки программных приложений на различных языках программирования, включая объектно-ориентированные языки;

– овладение студентами техникой тестирования и отладки приложений.

Основой для обучения программированию является курс информатики, изучаемый в школе. В свою очередь дисциплина является базовой для последующего специального обучения в области информационных технологий, разработки курсовых и дипломных проектов.

В результате изучения учебной дисциплины студент должен:

знать:

– основные понятия и принципы обработки информации, основы организации компьютерной обработки информации;

– принципы проектирования алгоритмов и их реализации;

– основные методы и средства эффективной разработки программного обеспечения;

– методы тестирования и отладки программ;

– принципы и технологии объектно-ориентированного программирования;

– объектно-ориентированные библиотеки, предназначенные для построения пользовательских приложений;

уметь:

– проектировать эффективные алгоритмы решения поставленной задачи;

– выбирать наиболее подходящие структуры данных, программные и технические средства реализации алгоритма;

– разрабатывать программные приложения, в том числе с объектно-ориентированным дизайном, с заданной функциональностью;

– использовать современные технологии разработки программ;

владеть:

– основными методами алгоритмизации практических задач;

– навыками разработки, тестирования и отладки программ в конкретных средах разработки.

В результате изучения учебной дисциплины «Программирование» формируются следующие компетенции:

профессиональные:

Специалист должен быть способен:

Проектно-конструкторская деятельность

– ПК-1. Проектировать, разрабатывать и тестировать программное обеспечение различных видов;

– ПК-2. Разрабатывать техническую документацию на программное обеспечение;

– ПК-5. Проектировать, разрабатывать, внедрять и тестировать насыщенные Интернет приложения;

– ПК-6. Проектировать, разрабатывать системы баз данных;

Научно-исследовательская деятельность

– ПК-7. Применять профессиональные знания и навыки для проведения научных исследований в области прикладной информатики;

– ПК-8. Разрабатывать и совершенствовать методы исследований в области информационных и телекоммуникационных систем;

– ПК-9. Работать с научно-технической информацией с использованием современных информационных технологий;

– ПК-10. Формулировать выводы и рекомендации по применению результатов научно-исследовательской работы;

– ПК-11. Пользоваться глобальными информационными ресурсами;

Эксплуатационная деятельность

– ПК-12. На основе технической документации выполнять внедрение и сопровождение программного обеспечения, в том числе разработанного сторонними организациями;

– ПК-14. На основе технической документации выполнять внедрение и сопровождение насыщенных Интернет приложений, в том числе разработанных сторонними организациями;

– ПК-15. Организовывать заполнение систем баз данных, в том числе разработанных сторонними организациями, на начальном этапе внедрения таких систем;

– ПК-16. Выполнять системное администрирование, администрирование баз данных, администрирование насыщенных Интернет приложений.

Форма получения высшего образования – дневная.

Учебная дисциплина изучается на 1 и 2 курсах в 1, 2 и 3 семестрах.

Общее количество часов по учебной дисциплине – 684, количество аудиторных часов – 340, количество зачетных единиц – 17,5.

Распределение аудиторного времени в 1 семестре: лекции – 52 часа, лабораторные занятия – 84 часа.

Распределение аудиторного времени во 2 семестре: лекции – 52 часа, лабораторные занятия – 84 часа.

Распределение аудиторного времени в 3 семестре: лекции – 28 часов, УСР – 6 часов, лабораторные занятия – 34 часа.

Форма отчётности во всех семестрах – зачет, экзамен.

СОДЕРЖАНИЕ УЧЕБНОГО МАТЕРИАЛА

Раздел 1. Разработка программ на Паскале

Тема 1.1. Введение в программирование

Понятия: число, символ, строка. Простейший ввод и форматированный вывод чисел, символов, строк. Арифметические операции над числами. Длина строки. Позиция символа в строке. Написание и исполнение программ в среде Turbo Pascal.

Тема 1.2. Базовые алгоритмы на одномерных массивах

Простейшая Паскаль-программа – пример. Понятие об операторах присваивания, цикла и условия. Базовые алгоритмы на одномерных массивах: суммирование, подсчет, поиск, поиск минимума и максимума.

Тема 1.3. Методика разработки алгоритмов и программ

Переформулировать условия задачи. Выяснить, что на входе, что на выходе. Составить тестовые примеры (не забыв о крайних случаях). Решить тестовые примеры вручную. Разработать алгоритм программы. Выполнить ручную прокрутку. Если есть ошибки, то переработать алгоритм. Написать текст программы.

Тема 1.4. Перенос базовых алгоритмов на двумерные массивы

Строка, столбец двумерного массива. Первая и вторая диагонали двумерного массива. Индексы в двумерном массиве. Циклы для ввода/вывода элементов массивов. Использование циклов для обработки двумерного массива. Использование циклов для обработки отдельных строк и столбцов двумерного массива. Использование циклов для обработки первой и второй диагоналей двумерного массива.

Тема 1.5. Простейшие задачи на координатной плоскости

Координаты точки на плоскости. Расстояние между двумя точками. Расстояния от одной до множества точек. Расстояние от каждой точки множества до всех других точек этого множества.

Тема 1.6. Примеры разработки алгоритмов и программ

Задачи на одномерном массиве: суммирование, подсчет, поиск минимального и максимального элементов, поиск элемента с заданным свойством. Комбинации и суперпозиции этих алгоритмов. Задачи на двумерном массиве. Задачи на разработку нестандартных алгоритмов.

Раздел 2. Язык программирования Паскаль

Тема 2.1. Встроенные типы данных

Целые типы: byte, word, integer, longint. Вещественные типы: real, single, double. Булевский тип boolean. Символьный тип char. Строковый тип string. Множественный тип set. Файловый тип. Операции над целыми числами: сложение, вычитание, умножение, DIV, MOD.

Тема 2.2. Встроенные функции обработки строк

Delete(a,b,c) – удалить из строки a символы с позиции b в количестве c штук. Pos(a,b) – выяснить позицию, с которой строка a встречается в строке b, Copy(a,b,c) – взять подстроку из строки a с позиции b, с символов. Insert(a,b,c) – вставить строку a в строку b с позиции c.

Тема 2.3. Пользовательские процедуры и функции

Функции и процедуры. Формальные параметры и фактические параметры. Глобальные переменные и локальные переменные. Области видимости переменных. Возвращаемые значения. Определение и вызов процедур и функций. Примеры.

Тема 2.4. Типы данных, определяемые программистом

Синтаксис и семантика. Примеры. Понятие о типах данных, определяемых программистом. Задачи о поиске прямой. Типы данных: точка, прямая, множество точек. Задача о множестве треугольников. Типы данных: отрезок, треугольник, множество отрезков, множество треугольников. Достоинства использования типов данных, определяемых программистом.

Тема 2.5. Встроенные функции преобразования типов

Функция Ord для преобразований символьных величин в целые. Функция Chr для преобразования целых величин в символьные. Процедура Val для преобразования строковых величин в числовые. Процедура Str для преобразования числовых величин в строковые. Функция Round для преобразования вещественных величин в целые округлением. Функция Int для преобразования вещественных величин в целые отбрасыванием дробной части.

Тема 2.6. Операторы организации циклов

Оператор for для циклов с переменной счета и фиксированным количеством итераций. Оператор while для циклов с предварительной проверкой условий завершения. Оператор repeat для циклов с пост-проверкой условий.

Оператор continue для перехода к следующей итерации цикла. Оператор break для досрочного выхода из ближайшего цикла.

Тема 2.7. Работа с файлами

Объявление файловых переменных. Текстовые файлы. Двоичные файлы. Входные и выходные файлы. Оператор assign для связывания файловой переменной с именем файла на диске. Процедура ReSet для открытия файла на чтения. Процедура ReWrite для открытия файла на запись. Процедура Close для закрытия файла.

Раздел 3. Разработка алгоритмов

Тема 3.1. Методы сортировки

Определение и примеры сортировки. Поиск минимального и максимального элемента с номером. Сортировка обменом. Проблемы сортировки обменом. Сортировка пузырьком. Достоинство сортировки пузырьком.

Оценка сложности сортировок обменом и пузырьком. Сортировка подсчетом. Оценка сложности сортировки подсчетом. Область применения сортировки подсчетом. Задачи на сортировку.

Тема 3.2. Структуры данных

Стек. Вершина стека. Операции: добавить в стек, взять из стека. Понятие о событиях: переполнение стека, пустой стек. Очередь. Примеры решения задач. Задача о шахматном коне. Задача о кусках. Задача о корректности последовательности скобок. Задача о лабиринте.

Тема 3.3. Задачи и алгоритмы на плоскости и в пространстве

Декартова система координат. Точки, отрезки, прямые. Уравнение прямой через две точки. Уравнение перпендикуляра к прямой через заданную точку. Треугольники, многоугольники. Вычисление площадей. Принадлежность точки фигуре. Минимальная выпуклая оболочка. Основные соотношения в треугольнике.

Тема 3.4. Рекурсия

Понятие о рекурсии. Задача «Ханойская башня». Примеры рекурсивного решения задач: нахождение наибольшего общего делителя, количество способов составления суммы M из N чисел. Правила написания рекурсивных программ. Рекурсивные процедуры и функции. Отладка рекурсивных процедур и функций.

Раздел 4. Технология программирования

Тема 4.1. Методы и средства разработки, тестирования и отладки программ

Средства разработки программ: текстовые редактор, компилятор, интегрированная среда отладки. Документирование программ: комментирование функциональных модулей, комментирование фрагментов исходных текстов. Стили форматирования исходных текстов программ. Тестирование и комплексная отладка.

Тема 4.2. Перспективы аппаратного и программного обеспечения ЭВМ

Сети ЭВМ. Локальные и глобальные сети. Обзор возможностей пользователя сети Internet. Информационно-справочные, аналитические и экспертные системы. Системы автоматизированного проектирования. Системное программное обеспечение.

Раздел 5. Язык программирования С

Тема 5.1. Базовые элементы языка программирования С

Основные этапы разработки программ на языке С. Алфавит языка, ключевые слова, идентификаторы, операторы, инструкции, разделяющие знаки. Переменные и константы. Комментарии и их использование.

Типы данных. Встроенные и пользовательские типы данных. Простые и

структурированные типы данных. Целочисленные типы данных. Представление в памяти целочисленных типов данных. Вещественные типы данных. Элементарный ввод-вывод.

Тема 5.2. Операции и выражения

Понятия операции и выражения. Операция присваивания. Множественное присваивание. Арифметические операции. Операции сравнения и логические операции. Приоритеты операций в выражениях. Преобразование типов в выражениях. Явное приведение типов.

Тема 5.3. Операторы управления вычислительным процессом

Реализация базовых алгоритмических структур средствами языка программирования – управляющие инструкции. Условный оператор if…else и if. Оператор множественного выбора switch.

Цикл while: особенности и примеры использования. Цикл do…while. Цикл со счетчиком for. Бесконечные циклы. Инструкции break и continue.

Тема 5.4. Адресная арифметика

Понятие указателя. Объявление указателя. Операторы взятия адреса и разыменования. Присваивание указателей. Нетипизированные указатели. Неявное и явное приведение типов указателей. Арифметические операции с указателями. Сравнение указателей. Указатели на указатели.

Тема 5.5. Массивы

Определение массива. Одномерные массивы. Размещение одномерных массивов в памяти. Объявление и инициализация массива. Двумерные и многомерные массивы. Доступ к элементам массивов: оператор индексирования.

Указатели на массивы. Доступ к элементам массивов с помощью указателей. Массивы указателей.

Базовые алгоритмы обработки массивов. Двоичный поиск.

Тема 5.6. Строки символов

Понятие строки в языке С. Объявление и инициализация строк. Указатели на строки и символы. Инициализация указателей на символы с помощью строковых констант. Доступ к элементам строки по индексу. Доступ к элементам строки с помощью указателя. Массивы строк.

Базовые алгоритмы обработки строк.

Тема 5.7. Функции

Определение функции. Объявление функции. Имя и тело функции. Параметры функции. Прототип функции. Механизмы передачи параметров в вызываемую функцию и из вызываемой функции. Передача параметров по значению.

Тема 5.8. Классы хранения и видимость переменных

Общие положения. Область определения и видимость идентификатора. Локальные и глобальные переменные, их области видимости. Автоматические переменные. Статические переменные. Класс хранения static.

Тема 5.9. Разработка и использование функций

Передача указателей в функции. Инициализация параметров функции. Возвращаемые значения. Возврат указателей. Функции, не возвращающие значения. Передача параметров в функцию main(). Примеры разработки и использования функций.

Тема 5.10. Рекурсивные функции

Виды рекурсии. Понятие рекурсивной функции. Примеры рекурсивных функций. Основные преимущества и недостатки применения рекурсивных функций. Правила разработки рекурсивных функций. Типичные ошибки при разработке рекурсивных функций. Отладка рекурсивных функций.

Тема 5.11. Библиотечные функции обработки и преобразования данных

Понятие библиотеки функций. Функции преобразования данных. Стандартные математические функции. Функции классификации и преобразования символов. Функции для работы с блоками памяти. Функции для работы со строками символов. Примеры работы с библиотечными функциями.

Тема 5.12. Работа с битами

Основные понятия. Двоичная система счисления. Побитовые операции. Типичные операции при работе с битами. Битовые маски. Примеры программ для работы с битами.

Тема 5.13. Структуры, объединения и перечисления

Общие положения. Шаблон структуры. Внешний и внутренний шаблоны. Структурные переменные. Правила выравнивания структурных переменных в памяти. Оператор typedef описания собственного типа данных. Вложенные структуры. Указатели на структуры. Доступ к членам структур с помощью указателей. Массивы структурных переменных. Использование структур в функциях. Объединения. Перечисления.

Тема 5.14. Препроцессор

Общие положения. Директивы #define и #undef. Обработка директив #define и #undef. Включение файлов. Директива #include. Условная компиляция. Директивы #if, #else, #elif.

Раздел 6. Управление ресурсами в языке С

Тема 6.1. Управление памятью

Статическое и динамическое распределение памяти. Функции динамического выделения и освобождения памяти. Оператор определения размера (sizeof). Динамическое выделение памяти для массивов и структур. Примеры на выделение и освобождение памяти.

Тема 6.2. Работа с файлами

Общие положения. Режимы доступа к файлам. Открытие и закрытие файла. Режимы открытия файлов. Доступ к файлам через поток ввода-

вывода. Понятие потока ввода-вывода. Посимвольный и построчный ввод вывод. Форматированный ввод-вывод.

Произвольный доступ к файлам. Блоковый ввод-вывод информации. Префиксный доступ к файлам. Функции префиксного файлового ввода-вывода. Управление указателем чтения-записи.

Тема 6.3. Связанные динамические структуры данных

Понятие динамической структуры данных. Структуры, ссылающиеся на себя. Связанные списки. Стеки. Очереди. Выделение и освобождение памяти под элементы динамических структур данных. Реализация динамических структур данных. Удаление динамических структур данных.

Тема 6.4. Списки, виды списков

Виды списков. Однонаправленные списки. Кольцевые списки. Двунаправленные списки. Реализация списков. Методы доступа к элементам списка. Удаление списка.

Тема 6.5. Обработка списков

Стандартные процедуры обработки списков: просмотр списка, поиск в списке, вставка нового элемента в список, удаление элемента из списка, сортировка списка.

Тема 6.6. Управление файлами, директориями и накопителями

Создание и уничтожение файла, директория. Управление текущим накопителем и директорием. Чтение содержимого директория. Поиск файлов. Удаление и переименование файлов. Создание резервных копий файлов. Определение существования файла или директория. Определение и установка параметров файла.

Раздел 7. Объектно-ориентированное программирование на языке C++

Тема 7.1. Расширения языка

Расширения языка C++, не связанные с применением объектно-ориентированного программирования: различия интерпретации операторов, описание данных и распределение памяти для данных, определение констант, определение агрегатных типов, использование переменных-ссылок, аргументы функций по умолчанию, встроенные (inline) функции, перегрузка функций.

Тема 7.2. Понятие и описание классов

Понятие классов в языке C++. Описание классов в языке C++. Члены класса – данные и функции (методы класса). Описание объектов классов. Динамическое создание и уничтожение объектов классов. Статические члены класса. Управление доступом к членам класса. Доступ к данным класса. Вызов методов класса. Указатель this. Принцип инкапсуляции.

Тема 7.3. Конструкторы и деструкторы классов

Понятие конструктора и деструктора класса. Назначение и порядок вызова конструкторов. Назначение и порядок вызова деструкторов. Использование конструкторов для инициализации членов класса. Использование списка инициализации в конструкторе.

Специальные виды конструкторов: конструкторы копирования и приведения. Использование специальных конструкторов.

Тема 7.4. Перегрузка операторов и операций

Понятие перегрузки операторов и операций. Назначение перегрузки. Перегрузка операторов с использованием глобальных функций. Перегрузка операторов с использованием методов классов. Примеры перегрузки операторов и операций.

Тема 7.5. Друзья класса

Понятие друзей класса. Свойства друзей класса. Доступ к данным и методам дружественного класса. Свойства и правила использования спецификатора friend. Использование дружественности при перегрузке операторов.

Тема 7.6. Наследование классов

Понятие наследования классов. Понятие производного класса. Простое и множественное наследование классов. Механизм наследования классов. Синтаксис описания наследования. Управление доступом к членам класса при наследовании. Вызов конструкторов и деструкторов при наследовании.

Тема 7.7. Полиморфизм и позднее связывание

Понятие полиморфизма в объектно-ориентированном программировании. Раннее и позднее связывание в C++. Понятие виртуальной функции.

Описание и использование виртуальных функций. Ограничения на использование виртуальных функций. Реализация механизма виртуальных функций.

Понятие и назначение абстрактного класса. Понятие чистой виртуальной функции. Свойства и правила использования абстрактных классов.

Тема 7.8. Описание и использование свойств классов

Понятие свойства класса. Описание свойств. Реализация методов доступа к свойствам. Использование свойств. Простые свойства. Определение и использование массивов свойств. Индексированные свойства. Свойства и иерархия классов. Ограничения, связанные со свойствами.

Тема 7.9. Описание и использование интерфейсов

Понятие интерфейса. Описание интерфейса. Иерархия наследования интерфейсов. Глобально-уникальная идентификация интерфейса. Реализация интерфейса классом. Совместимость классов и интерфейсов по типу.

Тема 7.10. Исключения

Обработка ошибок в языке C++. Проблема обработки ошибок. Общие сведения об исключениях. Определения типа исключения. Иерархия исключений. Получение дополнительной информации об исключении.

Спецификация функций, обрабатывающих исключения. Обработка исключений. Создание исключения, перехват исключения. Примеры обработки исключений. Исключения, которые не являются ошибками.

Раздел 8. Организация ввода-вывода в языке C++

Тема 8.1. Библиотека классов ввода-вывода

Управление вводом-выводом в языке C++. Классы потоков языка C++. Библиотека классов-потоков. Иерархия классов библиотеки потоков. Предопределенные объекты-потоки.

Ввод-вывод встроенных типов данных. Операции помещения в поток и извлечения из потока. Функции занесения информации в поток и получения информации из потока. Ввод-вывод пользовательских типов данных путем перекрытия операций помещения в поток и извлечения из потока.

Тема 8.2. Форматирование данных

Состояние потока. Форматирующие функции-элементы. Простые манипуляторы. Параметризованные манипуляторы. Форматирование данных при вводе и выводе. Функции форматирования. Флаги форматирования.

Тема 8.3. Файловый ввод-вывод

Потоки и файловый ввод-вывод. Файловый ввод-вывод с применением потоков. Классы для файлового ввода-вывода. Создание объектов – файловых потоков: с открытием и без открытия файла, с привязкой файла к существующему потоку. Функции позиционирования файлового потока.

Раздел 9. Шаблоны классов

Тема 9.1. Назначение и описание шаблонов классов и функций

Шаблоны классов в языке C++. Назначение шаблонов классов. Параметры шаблонов классов. Создание шаблонов для агрегатных типов данных. Описание классов с помощью шаблонов. Создание шаблонов функций. Методы использования шаблонов функций.

Тема 9.2. Стандартная библиотека шаблонов STL

Назначение стандартной библиотеки шаблонов STL. Организация библиотеки шаблонов STL. Пространство имен стандартной библиотеки шаблонов STL. Неименованные пространства имен.

Понятие итератора в STL. Операции с итераторами. Свойства итераторов. Обратные итераторы.

Назначение стандартного библиотечного класса string. Операции манипулирования строками. Примеры использования класса string.

Тема 9.3. Контейнерные классы в STL

Назначение контейнерных классов. Виды контейнерных классов: вектора, списки, множества, отображения. Конструкторы контейнерных классов. Описание объектов контейнерных классов. Основные методы контейнерных классов. Оценка сложности операций для контейнерных классов. Примеры использования контейнерных классов.

Тема 9.4. Основные алгоритмы STL

Обзор алгоритмов стандартной библиотеки STL. Алгоритмы, не модифицирующие последовательность. Алгоритмы, модифицирующие последовательность. Алгоритмы сортировки и поиска.

УЧЕБНО-МЕТОДИЧЕСКАЯ КАРТА УЧЕБНОЙ ДИСЦИПЛИНЫ

Номер раздела, темы	Название раздела, темы	Количество аудиторных часов					Количество часов УСР	Формы контроля знаний
		Лекции	Практические занятия	Семинарские занятия	Лабораторные занятия	Иное		
1	2	3	4	5	6	7	8	9
1	**Разработка программ на Паскале**	**24**	**0**	**0**	**36**	**0**	**0**	
1.1	*Введение в программирование* 1. Понятия: число, символ, строка 2. Простейший ввод и форматированный вывод чисел, символов, строк 3. Арифметические операции над числами 4. Написание и исполнение программ в среде TurboPascal	4	-	-	6	-	-	Защита отчётов по лабораторной работе
1.2	*Базовые алгоритмы на одномерных массивах* 1. Простейшая Паскаль-программа – пример 2. Понятие об операторах присваивания, цикла и условия 3. Базовые алгоритмы на одномерных массивах: суммирование, подсчет, поиск, поиск минимума и максимума	4	-	-	6	-	-	Защита отчётов по лабораторной работе
1.3	*Методика разработки алгоритмов и программ* 1. Переформулировать условия задачи 2. Составить тестовые примеры 3. Разработать алгоритм программы 4. Написать текст программы	4	-	-	6	-	-	Защита отчётов по лабораторной работе
1.4	*Перенос базовых алгоритмов на двумерные массивы* 1. Строка, столбец двумерного массива 2. Циклы для ввода/вывода элементов массива 3. Использование циклов для обработки двумерного массива	4	-	-	6	-	-	Защита отчётов по лабораторной работе
1.5	*Простейшие задачи на координатной плоскости* 1. Координаты точки на плоскости 2. Расстояние между двумя точками 3. Расстояния между соседними точками 4. Расстояния от одной до множества точек	4	-	-	6	-	-	Защита отчётов по лабораторной работе

1	2	3	4	5	6	7	8	9
1.6	*Примеры разработки алгоритмов и программ* 1. Задачи на одномерном массиве 2. Комбинации и суперпозиции алгоритмов для одномерных массивов 3. Задачи на двумерном массиве 4. Задачи на разработку нестандартных алгоритмов	4	-	-	6	-	-	Защита отчётов по лабораторной работе
2	**Язык программирования Паскаль**	**16**	**0**	**0**	**32**	**0**	**0**	
2.1	*Встроенные типы данных* 1. Целые типы: byte, word, integer, longint 2. Вещественные типы: real, single, double 3. Булевский тип Boolean 4. Операции над целыми числами: сложение, вычитание, умножение, DIV, MOD	4	-	-	6	-	-	Защита отчётов по лабораторной работе
2.2	*Встроенные функции обработки строк* 1. Delete(a,b,c) – удалить из строки a символы с позиции b в количестве c штук 2. Pos(a,b) – выяснить позицию, с которой строка a встречается в строке b 3. Copy(a,b,c) – взять подстроку из строки a с позиции b, с символов 4. Insert(a,b,c) – вставить строку a в строку b с позиции c	2	-	-	6	-	-	Защита отчётов по лабораторной работе
2.3	*Пользовательские процедуры и функции* 1. Функции и процедуры 2. Формальные параметры и фактические параметры 3. Глобальные переменные и локальные переменные 4. Области видимости переменных	2	-	-	4	-	-	Защита отчётов по лабораторной работе
2.4	*Типы данных, определяемые программистом* 1. Синтаксис и семантика 2. Понятие о типах данных, определяемых программистом 3. Типы данных 4. Достоинства использования типов данных, определяемых программистом	2	-	-	4	-	-	Защита отчётов по лабораторной работе
2.5	*Встроенные функции преобразования типов* 1. Функция Ord 2. Процедура Val 3. Функция Round 4. Функция Int	2	-	-	4	-	-	Защита отчётов по лабораторной работе
2.6	*Операторы организации циклов* 1. Оператор for для циклов с переменной счета и фиксированным количеством итераций 2. Оператор while для циклов с предварительной проверкой условий завершения 3. Оператор repeat для циклов с пост-проверкой условий 4. Оператор continue для перехода к следующей итерации цикла	2	-	-	4	-	-	Защита отчётов по лабораторной работе

1	2	3	4	5	6	7	8	9
2.7	*Работа с файлами* 1. Объявление файловых переменных 2. Текстовые файлы 3. Оператор assign для связывания файловой переменной с именем файла на диске 4. Процедура Close для закрытия файла	2	-	-	4	-	-	Защита отчётов по лабораторной работе
3	**Разработка алгоритмов**	**8**	**0**	**0**	**12**	**0**	**0**	
3.1	*Методы сортировки* 1. Определение и примеры сортировки 2. Поиск минимального и максимального элемента с номером 3. Оценка сложности сортировок обменом и пузырьком 4. Область применения сортировки подсчетом	2	-	-	4	-	-	Защита отчётов по лабораторной работе
3.2	*Структуры данных* 1. Стек 2. Операции: добавить в стек, взять из стека 3. Понятие о событиях: переполнение стека, пустой стек 4. Очередь. Примеры решения задач	2	-	-	4	-	-	Защита отчётов по лабораторной работе
3.3	*Задачи и алгоритмы на плоскости и в пространстве* 1. Декартова система координат 2. Точки, отрезки, прямые 3. Вычисление площадей 4. Принадлежность точки фигуре	2	-	-	2	-	-	Защита отчётов по лабораторной работе
3.4	*Рекурсия* 1. Понятие о рекурсии 2. Примеры рекурсивного решения задач: нахождение наибольшего общего делителя, количество способов составления суммы M из N чисел 3. Правила написания рекурсивных программ 4. Рекурсивные процедуры и функции	2	-	-	2	-	-	Защита отчётов по лабораторной работе
4	**Технология программирования**	**4**	**0**	**0**	**4**	**0**	**0**	
4.1	*Методы и средства разработки, тестирования и отладки программ* 1. Средства разработки программ: текстовый редактор, компилятор, интегрированная среда отладки 2. Документирование программ: комментирование функциональных модулей, комментирование фрагментов исходных текстов 3. Стили форматирования исходных текстов программ 4. Тестирование и комплексная отладка	2	-	-	2	-	-	Защита отчётов по лабораторной работе

1	2	3	4	5	6	7	8	9
4.2	*Перспективы аппаратного и программного обеспечения ЭВМ* 1. Сети ЭВМ 2. Обзор возможностей пользователя сети Internet 3. Системы автоматизированного проектирования 4. Системное программное обеспечение	2	-	-	2	-	-	Защита отчётов по лабораторной работе
								Зачет, экзамен
	Всего часов в 1 семестре:	**52**	**0**	**0**	**84**	**0**	**0**	
5	**Язык программирования С**	**36**	**0**	**0**	**54**	**0**	**0**	
5.1	*Базовые элементы языка программирования С* 1. Основные этапы разработки программ на языке С 2. Алфавит языка, ключевые слова, идентификаторы, операторы, инструкции, разделяющие знаки 3. Типы данных 4. Элементарный ввод-вывод	4	-	-	4	-	-	Защита лабораторной работы
5.2	*Операции и выражения* 1. Операция присваивания 2. Арифметические операции 3. Операции сравнения и логические операторы 4. Преобразования типов в выражениях	2	-	-	2	-	-	Защита лабораторной работы
5.3	*Операторы управления вычислительным процессом* 1. Условный оператор if…else и if 2. Цикл while: особенности и примеры использования 3. Цикл со счетчиком for 4. Операторы break и continue	2	-	-	4	-	-	Защита лабораторной работы
5.4	*Адресная арифметика* 1. Понятие указателя 2. Операторы взятия адреса и разыменования 3. Неявное и явное приведение типов указателей 4. Арифметические операции с указателями	2	-	-	4	-	-	Защита лабораторной работы
5.5	*Массивы* 1. Определение массива 2. Доступ к элементам массивов: оператор индексирования 3. Доступ к элементам массивов с помощью указателей 4. Базовые алгоритмы обработки массивов	4	-	-	8	-	-	Защита лабораторной работы

1	2	3	4	5	6	7	8	9
5.6	*Строки* 1. Понятие строки в языке С 2. Объявление и инициализация строк 3. Указатели на строки и символы 4. Базовые алгоритмы обработки строк	4	-	-	6	-	-	Защита лабораторной работы
5.7	*Функции* 1. Объявление функции 2. Параметры функции 3. Прототип функции 4. Механизмы передачи параметров в вызываемую функцию и из вызываемой функции	2	-	-	4	-	-	Защита лабораторной работы
5.8	*Классы хранения и видимость переменных* 1. Общие положения 2. Область определения и видимость идентификаторов 3. Локальные и глобальные переменные, их область видимости 4. Статические переменные	2	-	-	2	-	-	Защита лабораторной работы
5.9	*Разработка и использование функций* 1. Передача указателей в функции 2. Возвращаемые значения 3. Передача параметров в функцию main() 4. Примеры разработки и использования функций	2	-	-	6	-	-	Защита лабораторной работы
5.10	*Рекурсивные функции* 1. Понятие рекурсивной функции 2. Примеры рекурсивных функций 3. Правила разработки рекурсивных функций 4. Отладка рекурсивных функций	2	-	-	2	-	-	Защита лабораторной работы
5.11	*Библиотечные функции обработки и преобразования данных* 1. Понятие библиотеки функций 2. Стандартные математические функции 3. Функции для работы со строками символов 4. Примеры работы с библиотечными функциями	4	-	-	4	-	-	Защита лабораторной работы
5.12	*Работа с битами* 1. Двоичная система счисления 2. Побитовые операции 3. Типичные операции при работе с битами 4. Примеры программ для работы с битами	2	-	-	4	-	-	Защита лабораторной работы

1	2	3	4	5	6	7	8	9
5.13	*Структуры, объединения и перечисления* 1. Шаблон структуры 2. Структурные переменные 3. Указатели на структуры 4. Использование структур в функциях	2	-	-	2	-	-	Защита лабораторной работы
5.14	*Препроцессор* 1. Общие положения 2. Директивы #define и #undef 3. Включение файлов 4. Условная компиляция	2	-	-	2	-	-	Защита лабораторной работы
6	**Управление ресурсами в языке С**	**16**	**0**	**0**	**30**	**0**	**0**	
6.1	*Управление памятью* 1. Статическое и динамическое распределение памяти 2. Функции динамического выделения и освобождения памяти 3. Динамическое выделение памяти для массивов и структур 4. Примеры на выделение и освобождение памяти	2	-	-	4	-	-	Защита лабораторной работы
6.2	*Работа с файлами* 1. Открытие и закрытие файла 2. Доступ к файлам через поток ввода-вывода 3. Форматированный ввод-вывод 4. Префиксный доступ к файлам	4	-	-	8	-	-	Защита лабораторной работы
6.3	*Связанные динамические структуры данных* 1. Понятие динамической структуры данных 2. Структуры, ссылающиеся на себя 3. Выделение и освобождение памяти под элементы динамических структур данных 4. Реализация динамических структур данных	2	-	-	2	-	-	Защита лабораторной работы
6.4	*Списки, виды списков* 1. Виды списков 2. Реализация списков 3. Методы доступа к элементам списка 4. Удаление списка	4	-	-	6	-	-	Защита лабораторной работы
6.5	*Обработка списков* 1. Стандартные процедуры обработки списков 2. Поиск в списке 3. Вставка и удаление элементов списка 4. Сортировка списка	2	-	-	6	-	-	Защита лабораторной работы

1	2	3	4	5	6	7	8	9
6.6	*Управление файлами, директориями и накопителями* 1. Создание и уничтожение файла, директория 2. Управление текущим накопителем и директорием 3. Чтение содержимого директория 4. Определение и установка параметров файла	2	-	-	4	-	-	Защита лабораторной работы
								Зачет, экзамен
	Всего часов во 2 семестре:	**52**	**0**	**0**	**84**	**0**	**0**	
	Всего часов на 1 курсе:	**104**	**0**	**0**	**168**	**0**	**0**	
7	**Объектно-ориентированное программирование на языке С++**	**16**	**0**	**0**	**20**	**0**	**4**	
7.1	*Расширения языка* 1. Различия интерпретации операторов 2. Описание данных и распределение памяти для данных 3. Использование переменных-ссылок 4. Встроенные (inline) функции и перегрузка функций	2	-	-	2	-	-	Защита лабораторной работы
7.2	*Понятие и описание классов* 1. Понятие классов в языке С++ 2. Члены класса – данные и функции (методы класса) 3. Вызов методов класса и указатель this 4. Принцип инкапсуляции	2	-	-	2	-	-	Защита лабораторной работы
7.3	*Конструкторы и деструкторы классов* 1. Понятие конструктора и деструктора класса 2. Назначение и порядок вызова конструкторов и деструкторов 3. Специальные виды конструкторов 4. Использование специальных конструкторов	2	-	-	2	-	-	Защита лабораторной работы
7.4	*Перегрузка операторов и операций* 1. Понятие перегрузки операторов и операций 2. Назначение перегрузки 3. Примеры перегрузки операторов и операций	2	-	-	2	-	-	Защита лабораторной работы
7.5	*Друзья класса* 1. Понятие друзей класса 2. Свойства друзей класса 3. Доступ к данным и методам дружественного класса 4. Использование дружественности при перегрузке операторов	2	-	-	2	-	-	Защита лабораторной работы
7.6	*Наследование классов* 1. Простое наследование классов 2. Множественное наследование классов 3. Механизм наследования классов 4. Вызов конструкторов и деструкторов при наследовании	2	-	-	2	-	-	Защита лабораторной работы

1	2	3	4	5	6	7	8	9
7.7	*Полиморфизм и позднее связывание* 1. Понятие полиморфизма в объектно-ориентированном программировании 2. Раннее и позднее связывание в языке С++ 3. Описание и использование виртуальных функций 4. Понятие и назначение абстрактного класса	2	-	-	2	-	-	Защита лабораторной работы
7.8	*Описание и использование свойств классов* 1. Понятие свойства класса 2. Реализация методов доступа к свойствам 3. Использование свойств 4. Определение и использование массивов свойств	-	-	-	2	-	2	Защита лабораторной работы
7.9	*Описание и использование интерфейсов* 1. Понятие интерфейса 2. Описание интерфейса 3. Иерархия наследования интерфейсов 4. Реализация интерфейса классом	2	-	-	2	-	-	Защита лабораторной работы
7.10	*Исключения* 1. Общие сведения об исключениях 2. Определения типа исключения 3. Обработка исключений 4. Примеры обработки исключений	-	-	-	2	-	2	Защита лабораторной работы
8	**Организация ввода-вывода в языке С++**	**6**	**0**	**0**	**6**	**0**	**0**	
8.1	*Библиотека классов ввода-вывода* 1. Управление вводом-выводом в языке С++ 2. Классы потоков языка С++ 3. Библиотеки классов-потоков 4. Предопределенные объекты-потоки	2	-	-	2	-	-	Защита лабораторной работы
8.2	*Форматирование данных* 1. Форматирующие функции-элементы 2. Простые манипуляторы 3. Параметризованные манипуляторы 4. Форматирование данных при вводе и выводе	2	-	-	2	-	-	Защита лабораторной работы
8.3	*Файловый ввод-вывод* 1. Файловый ввод-вывод с применением потоков 2. Классы для файлового ввода-вывода 3. Создание объектов – файловых потоков 4. Функции позиционирования файлового потока	2	-	-	2	-	-	Защита лабораторной работы

1	2	3	4	5	6	7	8	9
9	**Шаблоны классов**	**6**	**0**	**0**	**8**	**0**	**2**	
9.1	*Назначение и описание шаблонов классов и функций* 1. Шаблоны классов в языке С++ 2. Описание классов с помощью шаблонов 3. Создание шаблонов функций 4. Методы использования шаблонов функций	2	-	-	2	-	-	Защита лабораторной работы
9.2	*Стандартная библиотека шаблонов STL* 1. Назначение стандартной библиотеки шаблонов STL 2. Пространство имен стандартной библиотеки шаблонов STL 3. Итераторы в STL 4. Назначение и использование стандартного библиотечного класса string	2	-	-	2	-	-	Защита лабораторной работы
9.3	*Контейнерные классы в STL* 1. Виды контейнерных классов: вектора, списки, множества, отображения 2. Конструкторы контейнерных классов 3. Основные методы контейнерных классов 4. Примеры использования контейнерных классов	2	-	-	2	-	-	Защита лабораторной работы
9.4	*Основные алгоритмы STL* 1. Обзор алгоритмов стандартной библиотеки STL 2. Алгоритмы, не модифицирующие последовательности 3. Алгоритмы, модифицирующие последовательности 4. Алгоритмы сортировки и поиска	-	-	-	2	-	2	Защита лабораторной работы
								Зачет, экзамен
	Всего часов в 3 семестре:	**28**	**0**	**0**	**34**	**0**	**6**	
	Всего часов на 2 курсе:	**28**	**0**	**0**	**34**	**0**	**6**	
	Всего часов:	**132**	**0**	**0**	**202**	**0**	**6**	

Доцент кафедры математических проблем управления и информатики, к.т.н., доцент М.С. Долинский

Доцент кафедры математических проблем управления и информатики, к.т.н., доцент В.А. Короткевич

Старший преподаватель кафедры математических проблем управления и информатики Л.И. Короткевич

ИНФОРМАЦИОННО-МЕТОДИЧЕСКАЯ ЧАСТЬ

Перечень тем лабораторных занятий

Раздел 1. Разработка программ на Паскале

1. Простые данные языка Паскаль и работа с ними
2. Операторы в Паскале
3. Суммирование элементов одномерного массива
4. Разработка алгоритмов программ и составление тестовых примеров
5. Циклы для ввода/вывода элементов массива
6. Использование циклов для обработки двумерного массива
7. Простейшие геометрические задачи
8. Комбинации и суперпозиции алгоритмов для одномерных массивов
9. Разработка нестандартных алгоритмов

Раздел 2. Язык программирования Паскаль

1. Операции над целыми числами
2. Обработка строк с помощью функций
3. Написание программы с помощью функций и процедур
4. Преобразование данных в Паскале
5. Решение задач с помощью циклов
6. Решение задач с входными файлами и выходными файлами

Раздел 3. Разработка алгоритмов

1. Применение сортировок
2. Решение задач на темы «Стек» и «Очередь»
3. Решение задач с помощью рекурсивных процедур и функций

Раздел 4. Технология программирования

1. Тестирование и комплексная отладка решений

Раздел 5. Язык программирования С

1. Типы данных
2. Элементарный ввод-вывод
3. Операции и выражения
4. Условный оператор
5. Циклы
6. Указатели
7. Арифметические операции с указателями
8. Массивы
9. Доступ к элементам массивов: оператор индексирования
10. Доступ к элементам массивов с помощью указателей
11. Базовые алгоритмы обработки массивов
12. Строки
13. Указатели на строки и символы
14. Базовые алгоритмы обработки строк
15. Функции
16. Механизмы передачи параметров

17. Классы хранения и видимость переменных
18. Передача указателей в функции
19. Передача параметров в функцию main()
20. Примеры разработки и использования функций
21. Рекурсивные функции
22. Библиотечные функции обработки и преобразования данных
23. Функции для работы со строками символов
24. Работа с битами
25. Типичные операции при работе с битами
26. Структуры, объединения и перечисления
27. Препроцессор

Раздел 6. Управление ресурсами в языке С

1. Управление памятью
2. Динамическое выделение памяти для массивов и структур
3. Работа с файлами
4. Доступ к файлам через поток ввода-вывода
5. Форматированный ввод-вывод
6. Префиксный доступ к файлам
7. Связанные динамические структуры данных
8. Списки, виды списков
9. Реализация списков
10. Методы доступа к элементам списка
11. Поиск в списках
12. Вставка и удаление элементов списка
13. Сортировка списков
14. Управление файлами, директориями и накопителями
15. Чтение содержимого директория

Раздел 7. Объектно-ориентированное программирование на языке С++

1. Расширения языка
2. Понятие и описание классов
3. Конструкторы и деструкторы классов
4. Перегрузка операторов и операций
5. Друзья класса
6. Наследование классов
7. Описание и использование виртуальных функций
8. Описание и использование свойств классов
9. Описание и использование интерфейсов
10. Исключения

Раздел 8. Организация ввода-вывода в языке С++

1. Библиотека классов ввода-вывода
2. Форматирование данных
3. Файловый ввод-вывод

Раздел 9. Шаблоны классов

1. Назначение и описание шаблонов классов и функций
2. Стандартная библиотека шаблонов STL
3. Контейнерные классы в STL
4. Основные алгоритмы STL

Диагностика компетенций студента

Учебным планом учреждения высшего образования по специальности в качестве формы текущей аттестации по учебной дисциплине «Программирование» предусмотрены зачеты и экзамены.

Для промежуточного контроля по учебной дисциплине и диагностики компетенций студента используются следующие формы:

- контрольные работы;
- тесты;
- контрольные вопросы;
- лабораторные работы.

Темы контрольных работ

1. Разработка программ на Паскале
2. Разработка алгоритмов
3. Операторы управления вычислительным процессом в языке С
4. Обработка символьной информации в языке С
5. Функции в языке С
6. Описание класса, методы класса, конструкторы и деструкторы
7. Перегрузка операторов и операций
8. Наследование классов
9. Шаблоны функций

Рекомендуемая литература

Основная

1. Долинский, М.С. Алгоритмизация и программирование на Turbo Pascal от простых до олимпиадных задач / М.С. Долинский. – СПб.: Питер, 2005. – 237 с.

2. Долинский, М.С. Решение сложных и олимпиадных задач по программированию: учебное пособие для школьников, студентов и преподавателей / М.С. Долинский. – СПб.: Питер, 2006. – 366 с.

3. Рапаков, Г.Г. Turbo Pascal для студентов и школьников / Г.Г. Рапаков, С.Ю. Ржеуцкая. – СПб.: БХВ-Петербург, 2012. – 352 с.

4. Фаронов, В.В. Turbo Pascal: учебное пособие для студентов вузов / В.В. Фаронов. – СПб.: Питер, 2010. – 368 с.

5. Огнева, М.В. Turbo Pascal: Первые шаги. Примеры и упражнения / М.В. Огнева, Е.В. Кудрина. – М.: Научная книга, 2008. – 100 с.

6. Моргун, А.Н. Программирование на языке Паскаль. Основы обработки структур данных / А.Н. Моргун, И.А. Кривель. – М.: Вильямс, 2006. – 576 с.

7. Павловская, Т.К. Паскаль. Программирование на языке высокого уровня / Т.К. Павловская. – СПб.: Питер, 2010. – 464 с.

8. Касаткин, А.И. Профессиональное программирование на языке Си: От Turbo C к Borland C++ / А.И. Касаткин, А.Н. Вальвачев. – Мн: Выш. шк., 1992. – 378 с.

9. Касаткин, А.И. Профессиональное программирование на языке Си: Управление ресурсами / А.И. Касаткин. – Мн: Выш. шк., 1992. – 432 с.

10. Керниган, Б. Язык программирования С / Б. Керниган, Д. Ритчи. – М.: Вильямс, 2019. – 288 с.

11. Шилд, Г. С. Полное руководство. Классическое издание / Г. Шилд. – М.: Вильямс, 2020. – 704 с.

12. Прата, С. Язык программирования С. Лекции и упражнения / С. Прата. – М.: Вильямс, 2018. – 928 с.

13. Прата, С. Язык программирования С++. Лекции и упражнения / С. Прата. – М.: Вильямс, 2018. – 1248 с.

14. Павловская, Т.К. С/С++. Процедурное и объектно-ориентированное программирование / Т.К. Павловская. – СПб.: Питер, 2018. – 496 с.

15. Страуструп, Б. Язык программирования С++. Специальное издание / Б. Страуструп. – М.: Бином, 2017. – 1136 с.

16. Страуструп, Б. Программирование. Принципы и практика использования С++ / Б. Страуструп. – М.: Вильямс, 2018. – 1328 с.

17. Хорев, П. Б. Объектно-ориентированное программирование / П. Б. Хорев. – М.: Academia, 2012. – 447 с.

18. Шилд, Г. С++. Базовый курс / Г. Шилд. – М.: Вильямс, 2018. – 624 с.

19. Галовиц, Я. С++17 STL. Стандартная библиотека шаблонов / Я. Галловиц. – СПб.: Питер, 2018. – 432 с.

20. STL – стандартная библиотека шаблонов С++ / П. Плаугер [и др.]. – СПб.: БХВ-Петербург, 2004. – 656 с.

21. Мюссер, Д. С++ и STL: справочное руководство / Д. Мюссер, Ж. Дердж, А. Сейни. – М.: Вильямс, 2010. – 432 с.

Дополнительная

1. Фаронов, В.В. Turbo Pascal 7.0. Учебный курс / В.В. Фаронов. – М.: Кнорус, 2012. – 368 с.

2. Гусева, А.И. Учимся программировать: Pascal 7.0 / А.И. Гусева. – М.: Диалог-МИФИ, 2012. – 256 с.

3. Лукин, С.Н. Турбо-Паскаль 7.0. Самоучитель для начинающих / С.Н. Лукин. – М.: Диалог-МИФИ, 2015. – 384 с.

4. Иванова, Г.С. Программирование: учебник для студентов вузов, обучающихся по направлению «Информатика и вычислительная техника» / Г.С. Иванова. – М.: КНОРУС, 2017. – 426 с.

5. Гавриков, М.М. Теоретические основы разработки и реализации языков программирования: учебное пособие / М.М. Гавриков. – М.: КНОРУС, 2013. – 355 с.

6. Гриффитс, Д. Изучаем программирование на С / Д. Гриффитс, Д. Гриффитс. – М.: ЭКСМО, 2013. – 624 с.

7. Дейтел, П. Как программировать на С / П. Дейтел, Х. Дейтел. – М.: Бином, 2017. – 1008 с.

8. Шилд, Г. С++. Полное руководство / Г. Шилд. – М.: Вильямс, 2019. – 800 с.

9. Павловская, Т.К. С/С++. Структурное программирование: практикум / Т.К. Павловская. – СПб.: Питер, 2007. – 239 с.

10. Немцова, Т. И. Программирование на языке высокого уровня. Программирование на языке С++. Учеб. пособие / Т. И. Немцова, С. Ю. Голова, А.И. Терентьев. – М.: Форум:ИНФА-М, 2012. – 512 с.

11. Лафоре, Р. Объектно-ориентированное программирование в С++ / Р. Лафоре. – СПб.: Питер, 2018. – 928 с.

12. Мейерс, С. Эффективный и современный С++. 42 рекомендации по использованию С++11 и С++14 / С. Мейерс. – М.: Вильямс, 2019. – 304 с.

13. Макаровских, Т.А. Языки и методы программирования. Создание простых GUI-приложений с помощью Visual C++ / Т.А. Макаровских. – М.: Ленанд, 2018. – 140 с.

14. Москвин, П.В. Азбука STL / П.В. Москвин. – М.: Телеком, 2012. – 262 с.

Методические рекомендации по организации и выполнению управляемой самостоятельной работы студентов

Для самостоятельного изучения выделяются следующие темы:

- описание и использование свойств классов;
- исключения;
- основные алгоритмы STL.

Тема 7.8. Описание и использование свойств классов – 2 часа

Цели: 1) овладеть знаниями по данной теме, терминологией и методологией; 2) сформировать компетенцию в умении описывать и использовать свойства классов.

Виды заданий УСР по теме с учетом модулей сложности:

А) *Задания, формирующие знания по учебному материалу на уровне узнавания:*

1. Соотнесите термины с определениями.
2. Исправьте ошибки в определениях.
3. Вставьте в определение соответствующий термин.

Форма выполнения заданий – индивидуальная.

Форма контроля выполнения заданий – тест.

Б) *Задания, формирующие компетенции на уровне воспроизведения*:

1. Дайте определения терминам.
2. Приведите примеры, подтверждающие или опровергающие правильность утверждений.

3. Сформулируйте основные возможности и преимущества использования свойств классов.
4. Приведите примеры использования свойств классов.

Форма выполнения заданий – индивидуальная.

Форма контроля выполнения заданий – тест, контрольные вопросы.

В) *Задания, формирующие компетенции на уровне применение полученных знаний:*

1. Обоснуйте преимущества и недостатки при использовании массивов свойств.
2. Обозначьте основные отличия в использовании простых и индексированных свойств.
3. Продемонстрируйте ограничения, связанные со свойствами и их использованием.
4. Напишите класс, в котором опишите несколько свойств и реализуйте методы доступа к этим свойствам.

Форма выполнения заданий – индивидуальная.

Форма контроля выполнения заданий – лабораторная работа.

Учебно-методическое обеспечение:

1) Рекомендуемая основная и дополнительная литература.

2) Конспект лекций по дисциплине.

3) Информация в сети Интернет.

Тема 7.10. Исключения – 2 часа

Цели: 1) овладеть знаниями по данной теме, терминологией и методологией; 2) сформировать компетенцию в умении использовать механизм исключений для перехвата и обработки ошибок.

Виды заданий УСР по теме с учетом модулей сложности:

А) *Задания, формирующие знания по учебному материалу на уровне узнавания:*

1. Соотнесите термины с определениями.
2. Исправьте ошибки в определениях.
3. Вставьте в определение соответствующий термин.

Форма выполнения заданий – индивидуальная.

Форма контроля выполнения заданий – тест.

Б) *Задания, формирующие компетенции на уровне воспроизведения*:

1. Дайте определения терминам.
2. Приведите примеры, подтверждающие или опровергающие правильность утверждений.
3. Сформулируйте основные проблемы, возникающие при обработке ошибок.
4. Дайте сравнительную характеристику различных типов исключений.
5. Сформулируйте основные принципы обработки исключений.
6. Приведите примеры использования исключений.

Форма выполнения заданий – индивидуальная.

Форма контроля выполнения заданий – тест, контрольные вопросы.

В) *Задания, формирующие компетенции на уровне применение полученных знаний:*

1. Обоснуйте выбор подходящего типа исключения.
2. Продемонстрируйте механизм перехвата исключений.
3. Продемонстрируйте получение дополнительной информации об исключениях.
4. Реализуйте функции, обрабатывающие исключения.
5. Продемонстрируйте использование механизма исключений для обработки ошибок.

Форма выполнения заданий – индивидуальная.

Форма контроля выполнения заданий – лабораторная работа.

Учебно-методическое обеспечение:

1) Рекомендуемая основная и дополнительная литература.

2) Конспект лекций по дисциплине.

3) Информация в сети Интернет.

Тема 9.4. Основные алгоритмы STL – 2 часа

Цели: 1) овладеть знаниями по данной теме, терминологией и методологией; 2) сформировать компетенцию в умении использовать основные алгоритмы STL при работе с контейнерными классами.

Виды заданий УСР по теме с учетом модулей сложности:

А) *Задания, формирующие знания по учебному материалу на уровне узнавания:*

1. Соотнесите термины с определениями.
2. Исправьте ошибки в определениях.
3. Вставьте в определение соответствующий термин.

Форма выполнения заданий – индивидуальная.

Форма контроля выполнения заданий – тест.

Б) *Задания, формирующие компетенции на уровне воспроизведения*:

1. Дайте определения терминам.
2. Приведите примеры, подтверждающие или опровергающие правильность утверждений.
3. Сформулируйте основные возможности и преимущества использования алгоритмов STL.
4. Дайте сравнительную характеристику различных алгоритмов.
5. Приведите примеры использования алгоритмов для различных контейнерных классов.

Форма выполнения заданий – индивидуальная.

Форма контроля выполнения заданий – тест, контрольные вопросы.

В) *Задания, формирующие компетенции на уровне применение полученных знаний:*

1. Обоснуйте выбор подходящего алгоритма STL.

2. Обозначьте основные отличия в сложности работы различных алгоритмов.
3. Продемонстрируйте использование алгоритмов для различных контейнерных классов.

Форма выполнения заданий – индивидуальная.

Форма контроля выполнения заданий – лабораторная работа.

Учебно-методическое обеспечение:

1) Рекомендуемая основная и дополнительная литература.
2) Конспект лекций по дисциплине.
3) Информация в сети Интернет.

Модульно-рейтинговая система оценки знаний студентов

Таблица рейтингового оценивания учебной деятельности студентов по дисциплине «Программирование» во 2 семестре

№	Форма контроля		Оценивание	Весовой коэффициент	Пояснение
1	2	3	4	5	6
1	Выполнение лабораторных работ (общее число работ N=23)	O1	взвешенная средняя оценка по 10-балльной шкале с двумя знаками после запятой	K1 = 0.6	каждая лабораторная работа имеет свой весовой коэффициент, который учитывает ее сложность
2	Выполнение текущих тестов на компьютере (общее число тестов N=7)	O2	средняя оценка по 10-балльной шкале с двумя знаками после запятой	K2 = 0.1	в каждом тесте 10 заданий
3	Контрольные работы (общее число работ N=2)	O3	взвешенная средняя оценка по 10-балльной шкале с двумя знаками после запятой	K3 = 0.1	каждая контрольная работа имеет свой весовой коэффициент, который учитывает ее сложность
4	Оценка преподавателя	O4	взвешенная средняя оценка по 10-балльной шкале с двумя знаками после запятой	K4 = 0.2	каждый учебный месяц имеет свой весовой коэффициент, выставляется с учетом: 1) выполнения учебной программы; 2) регулярности работы; 3) активности на лабораторных занятиях; 4) учебной дисциплины; 5) пропусков учебных занятий без уважительных причин

1	2	3	4	5	6
5	Итого за работу в семестре (промежуточный контроль)	ОС	оценка по 10-бальной шкале с двумя знаками после запятой	КС = 0.9	сумма оценок за каждую форму контроля, умноженных на соответствующий весовой коэффициент: ОС = К1*О1 + К2*О2 + К3*О3 + К4*О4
6	Зачет (итоговый контроль)				выставляется при ОС ≥ 4.00
7	Оценка на экзамене (итоговый контроль)	ОЭ	оценка по 100-бальной шкале с двумя знаками после запятой	КЭ = 0.01	допуск к экзамену при ОС ≥ 4.00, выполнение теста на компьютере
8	Итоговая оценка	ОИ	целая оценка по 10-балльной шкале		выставляется при ОЭ ≥ 40.00, ОИ = КС*ОС + КЭ*ОЭ

Примечание:

- все оценки, кроме итоговой, округляются до двух знаков после запятой по математическим правилам округления;
- итоговая оценка округляется до целого по математическим правилам округления.

Таблица рейтингового оценивания учебной деятельности студентов по дисциплине «Программирование» в 3 семестре

№	Форма контроля		Оценивание	Весовой коэффициент	Пояснение
1	2	3	4	5	6
1	Выполнение лабораторных работ группы 1 (общее число работ N=2)	О1	взвешенная средняя оценка по 10-балльной шкале с двумя знаками после запятой	К1 = 0.04	каждая лабораторная работа имеет свой весовой коэффициент, который учитывает ее сложность
2	Выполнение лабораторных работ группы 2 (общее число работ N=10)	О2	взвешенная средняя оценка по 10-балльной шкале с двумя знаками после запятой	К2 = 0.6	каждая лабораторная работа имеет свой весовой коэффициент, который учитывает ее сложность
3	Выполнение лабораторных работ группы 3 (общее число работ N=3)	О3	взвешенная средняя оценка по 10-балльной шкале с двумя знаками после запятой	К3 = 0.06	каждая лабораторная работа имеет свой весовой коэффициент, который учитывает ее сложность
4	Контрольные работы (общее число работ N=3)	О4	взвешенная средняя оценка по 10-балльной шкале с двумя знаками после запятой	К4 = 0.1	каждая контрольная работа имеет свой весовой коэффициент, который учитывает ее сложность

1	2	3	4	5	6
5	Оценка преподавателя	О5	взвешенная средняя оценка по 10-балльной шкале с двумя знаками после запятой	К5 = 0.2	каждый учебный месяц имеет свой весовой коэффициент, выставляется с учетом: 1) выполнения учебной программы; 2) регулярности работы; 3) активности на лабораторных занятиях; 4) учебной дисциплины; 5) пропусков учебных занятий без уважительных причин
6	Итого за работу в семестре (промежуточный контроль)	ОС	оценка по 10-бальной шкале с двумя знаками после запятой	КС = 0.9	сумма оценок за каждую форму контроля, умноженных на соответствующий весовой коэффициент: ОС = К1*О1 + К2*О2 + К3*О3 + К4*О4 + К5*О5
7	Зачет (итоговый контроль)				выставляется при ОС ≥ 4.00
8	Оценка на экзамене (итоговый контроль)	ОЭ	оценка по 100-бальной шкале с двумя знаками после запятой	КЭ = 0.01	допуск к экзамену при ОС ≥ 4.00, выполнение теста на компьютере
9	Итоговая оценка	ОИ	целая оценка по 10-балльной шкале		выставляется при ОЭ ≥ 40.00, ОИ = КС*ОС + КЭ*ОЭ

Примечание:

- все оценки, кроме итоговой, округляются до двух знаков после запятой по математическим правилам округления;
- итоговая оценка округляется до целого по математическим правилам округления.

4.2. ПЕРЕЧЕНЬ РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ ПО ДИСЦИПЛИНЕ «ПРОГРАММИРОВАНИЕ» (часть 2)

Основная

1. Касаткин, А.И. Профессиональное программирование на языке Си: От Turbo C к Borland C++ / А.И. Касаткин, А.Н. Вальвачев. – Мн: Выш. шк., 1992. – 378 с.

2. Касаткин, А.И. Профессиональное программирование на языке Си: Управление ресурсами / А.И. Касаткин. – Мн: Выш. шк., 1992. – 432 с.

3. Керниган, Б. Язык программирования С / Б. Керниган, Д. Ритчи. – М.: Вильямс, 2019. – 288 с.

4. Шилд, Г. С. Полное руководство. Классическое издание / Г. Шилд. – М.: Вильямс, 2020. – 704 с.

5. Прата, С. Язык программирования С. Лекции и упражнения / С. Прата. – М.: Вильямс, 2018. – 928 с.

6. Прата, С. Язык программирования С++. Лекции и упражнения / С. Прата. – М.: Вильямс, 2018. – 1248 с.

Дополнительная

1. Гриффитс, Д. Изучаем программирование на С / Д. Гриффитс, Д. Гриффитс. – М.: ЭКСМО, 2013. – 624 с.

2. Дейтел, П. Как программировать на С / П. Дейтел, Х. Дейтел. – М.: Бином, 2017. – 1008 с.