Приложение 1 к РПД Программирование
09.03.02 Информационные системы и технологии
Направленность (профиль) – Программно-аппаратные комплексы
Форма обучения – очная
Год набора - 2020

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ (МОДУЛЯ)

1.	Кафедра	Информатики и вычислительной техники
2.	Направление подготовки	09.03.02 Информационные системы и технологии
3.	Направленность (профиль)	Программно-аппаратные комплексы
4.	Дисциплина (модуль)	Программирование
5.	Форма обучения	очная
6.	Год набора	2020

1. Методические рекомендации

Приступая к изучению дисциплины, обучающемуся необходимо внимательно ознакомиться с тематическим планом занятий, списком рекомендованной литературы. Следует уяснить последовательность выполнения индивидуальных учебных заданий. Самостоятельная работа обучающегося предполагает работу с научной и учебной литературой, умение создавать тексты. Уровень и глубина усвоения дисциплины зависят от активной и систематической работы на лекциях, изучения рекомендованной литературы, выполнения контрольных письменных заданий.

При изучении дисциплины обучающиеся выполняют следующие задания:

- изучают рекомендованную научно-практическую и учебную литературу;
- выполняют задания, предусмотренные для самостоятельной работы.

Основными видами аудиторной работы обучающихся являются лекции и лабораторные занятия.

1.1. Методические рекомендации по организации работы обучающихся во время проведения лекционных занятий

В ходе лекций преподаватель излагает и разъясняет основные, наиболее сложные понятия темы, а также связанные с ней теоретические и практические проблемы, дает рекомендации на семинарское занятие и указания на самостоятельную работу.

Знакомство с дисциплиной происходит уже на первой лекции, где от обучающегося требуется не просто внимание, но и самостоятельное оформление конспекта. При работе с конспектом лекций необходимо учитывать тот фактор, что одни лекции дают ответы на конкретные вопросы темы, другие – лишь выявляют взаимосвязи между явлениями, помогая обучающемуся понять глубинные процессы развития изучаемого предмета как в истории, так и в настоящее время.

Конспектирование лекций – сложный вид вузовской аудиторной работы, предполагающий интенсивную умственную деятельность обучающегося. Конспект является полезным тогда, когда записано самое существенное и сделано это самим обучающимся. Не надо стремиться записать дословно всю лекцию. Такое

«конспектирование» приносит больше вреда, чем пользы. Целесообразно вначале понять основную мысль, излагаемую лектором, а затем записать ее. Желательно запись осуществлять на одной странице листа или оставляя поля, на которых позднее, при самостоятельной работе с конспектом, можно сделать дополнительные записи, отметить непонятные места.

Конспект лекции лучше подразделять на пункты, соблюдая красную строку. Этому в большой степени будут способствовать вопросы плана лекции, предложенные преподавателям. Следует обращать внимание на акценты, выводы, которые делает лектор, отмечая наиболее важные моменты в лекционном материале замечаниями «важно», «хорошо запомнить» и т.п. Можно делать это и с помощью разноцветных маркеров или ручек, подчеркивая термины и определения.

Целесообразно разработать собственную систему сокращений, аббревиатур и символов. Однако при дальнейшей работе с конспектом символы лучше заменить обычными словами для быстрого зрительного восприятия текста.

Работая над конспектом лекций, всегда необходимо использовать не только учебник, но и ту литературу, которую дополнительно рекомендовал лектор. Именно такая серьезная, кропотливая работа с лекционным материалом позволит глубоко овладеть теоретическим материалом.

1.2. Методические рекомендации по подготовке к лабораторным занятиям (семинарам)

Подготовку к каждому практическому занятию студент должен начать с ознакомления с планом практического занятия, который отражает содержание предложенной темы. Тщательное продумывание и изучение вопросов плана основывается на проработке текущего материала лекции, а затем изучения обязательной и дополнительной литературы, рекомендованной к данной теме. Все новые понятия по изучаемой теме необходимо выучить наизусть и внести в глоссарий, который целесообразно вести с самого начала изучения курса.

Результат такой работы должен проявиться в способности студента свободно ответить на теоретические вопросы практикума, его выступлении и участии в коллективном обсуждении вопросов изучаемой темы, правильном выполнении практических заданий и контрольных работ.

В процессе подготовки к практическим занятиям, студентам необходимо обратить особое внимание на самостоятельное изучение рекомендованной литературы. При всей полноте конспектирования лекции в ней невозможно изложить весь материал из-за лимита аудиторных часов. Поэтому самостоятельная работа с учебниками, учебными пособиями, научной, справочной литературой, материалами периодических изданий и Интернета является наиболее эффективным методом получения дополнительных знаний, позволяет значительно активизировать процесс овладения информацией, способствует более глубокому усвоению изучаемого материала, формирует у студентов свое отношение к конкретной проблеме.

Семинарские занятия завершают изучение наиболее важных тем учебной дисциплины. Они служат для углубленного изучения материала, а также для контроля преподавателем степени подготовленности студентов по изучаемой дисциплине.

Семинар предполагает решение и обсуждение математических задач по пройденным темам. На семинаре рассматриваются различные методы решения задач, ведется обсуждение результатов. В ходе семинарских занятий может осуществлять текущий контроль знаний в виде тестовых заданий.

При подготовке к семинару студенты имеют возможность воспользоваться консультациями преподавателя. Качество учебной работы студентов преподаватель оценивает с использованием технологической карты дисциплины, размещенной на сайте филиала МАГУ.

1.3. Методические рекомендации по работе с литературой

Работу с литературой целесообразно начать с изучения общих работ по теме, а также учебников и учебных пособий. Далее рекомендуется перейти к анализу монографий и статей, рассматривающих отдельные аспекты проблем, изучаемых в рамках курса, а также официальных материалов и неопубликованных документов (научно-исследовательские работы, диссертации), в которых могут содержаться основные вопросы изучаемой проблемы.

Работу с источниками надо начинать с ознакомительного чтения, т.е. просмотреть текст, выделяя его структурные единицы. При ознакомительном чтении закладками отмечаются те страницы, которые требуют более внимательного изучения.

В зависимости от результатов ознакомительного чтения выбирается дальнейший способ работы с источником. Если для разрешения поставленной задачи требуется изучение некоторых фрагментов текста, то используется метод выборочного чтения. Если в книге нет подробного оглавления, следует обратить внимание ученика на предметные и именные указатели.

Избранные фрагменты или весь текст (если он целиком имеет отношение к теме) требуют вдумчивого, неторопливого чтения с «мысленной проработкой» материала. Такое чтение предполагает выделение: 1) главного в тексте; 2) основных аргументов; 3) выводов. Особое внимание следует обратить на то, вытекает тезис из аргументов или нет.

Необходимо также проанализировать, какие из утверждений автора носят проблематичный, гипотетический характер и уловить скрытые вопросы.

Понятно, что умение таким образом работать с текстом приходит далеко не сразу. Наилучший способ научиться выделять главное в тексте, улавливать проблематичный характер утверждений, давать оценку авторской позиции – это сравнительное чтение, в ходе которого обучающийся знакомится с различными мнениями по одному и тому же вопросу, сравнивает весомость и доказательность аргументов сторон и делает вывод о наибольшей убедительности той или иной позиции.

Если в литературе встречаются разные точки зрения по тому или иному вопросу из-за сложности прошедших событий и правовых явлений, нельзя их отвергать, не разобравшись. При наличии расхождений между авторами необходимо найти рациональное зерно у каждого из них, что позволит глубже усвоить предмет изучения и более критично оценивать изучаемые вопросы. Знакомясь с особыми позициями авторов, нужно определять их схожие суждения, аргументы, выводы, а затем сравнивать их между собой и применять из них ту, которая более убедительна.

Следующим этапом работы с литературными источниками является создание конспектов, фиксирующих основные тезисы и аргументы. Можно делать записи на отдельных листах, которые потом легко систематизировать по отдельным темам изучаемого курса. Другой способ – это ведение тематических тетрадей-конспектов по одной какой-либо теме. Большие специальные работы монографического характера целесообразно конспектировать в отдельных тетрадях. Здесь важно вспомнить, что конспекты пишутся на одной стороне листа, с полями и достаточным для исправления и ремарок межстрочным расстоянием (эти правила соблюдаются для удобства редактирования). Если в конспектах приводятся цитаты, то непременно должно быть дано указание на источник (автор, название, выходные данные, № страницы). Впоследствии эта информации может быть использована при написании текста реферата или другого задания.

Таким образом, при работе с источниками и литературой важно уметь:

- сопоставлять, сравнивать, классифицировать, группировать, систематизировать информацию в соответствии с определенной учебной задачей;
- обобщать полученную информацию, оценивать прослушанное и прочитанное;

- фиксировать основное содержание сообщений; формулировать, устно и письменно, основную идею сообщения; составлять план, формулировать тезисы;
- готовить и презентовать развернутые сообщения типа доклада;
- работать в разных режимах (индивидуально, в паре, в группе), взаимодействуя друг с другом;
- пользоваться реферативными и справочными материалами;
- контролировать свои действия и действия своих товарищей, объективно оценивать свои действия;
- обращаться за помощью, дополнительными разъяснениями к преподавателю, другим обучающимся.
- пользоваться лингвистической или контекстуальной догадкой, словарями различного характера, различного рода подсказками, опорами в тексте (ключевые слова, структура текста, предваряющая информация и др.);
- использовать при говорении и письме перифраз, синонимичные средства, слова-описания общих понятий, разъяснения, примеры, толкования, «словотворчество»;
- повторять или перефразировать реплику собеседника в подтверждении понимания его высказывания или вопроса;
- обратиться за помощью к собеседнику (уточнить вопрос, переспросить и др.);
- использовать мимику, жесты (вообще и в тех случаях, когда языковых средств не хватает для выражения тех или иных коммуникативных намерений).

1.4. Методические рекомендации по подготовке к сдаче экзамена

Подготовка к экзамену способствует закреплению, углублению и обобщению знаний, получаемых, в процессе обучения, а также применению их к решению практических задач. Готовясь к экзамену, обучающийся ликвидирует имеющиеся пробелы в знаниях, углубляет, систематизирует и упорядочивает свои знания. На экзамене обучающийся демонстрирует то, что он приобрел в процессе изучения дисциплины.

В условиях применяемой в МАГУ балльно-рейтинговой системы подготовка к экзамену включает в себя самостоятельную и аудиторную работу обучающегося в течение всего периода изучения дисциплины и непосредственную подготовку в дни, предшествующие экзамену по разделам и темам дисциплины.

При подготовке к экзамену обучающимся целесообразно использовать не только материалы лекций, а и рекомендованные основную и дополнительную литературу.

При подготовке к промежуточной аттестации целесообразно:

- внимательно изучить перечень вопросов и определить, в каких источниках находятся сведения, необходимые для ответа на них;
- внимательно прочитать рекомендованную литературу;
- составить краткие конспекты ответов (планы ответов).

Качество учебной работы обучающихся преподаватель оценивает с использованием технологической карты дисциплины, размещенной на сайте филиала МАГУ.

1.5. Методические рекомендации по составлению глоссария

1. Внимательно прочитайте и ознакомьтесь с текстом. Вы встретите в нем много различных терминов, которые имеются по данной теме.
2. После того, как вы определили наиболее часто встречающиеся термины, вы должны составить из них список. Слова в этом списке должны быть расположены в строго алфавитном порядке, так как глоссарий представляет собой не что иное, как словарь специализированных терминов.
3. После этого начинается работа по составлению статей глоссария. Статья глоссария - это определение термина. Она состоит из двух частей: 1. точная формулировка

термина в именительном падеже; 2. содержательная часть, объемно раскрывающая смысл данного термина.

При составлении глоссария важно придерживаться следующих правил:

- стремитесь к максимальной точности и достоверности информации;
- старайтесь указывать корректные научные термины и избегать всякого рода жаргонизмов. В случае употребления такового, давайте ему краткое и понятное пояснение;
- излагая несколько точек зрения в статье по поводу спорного вопроса, не принимайте ни одну из указанных позиций. Глоссарий - это всего лишь констатация имеющихся фактов;
- также не забывайте приводить в пример контекст, в котором может употреблять данный термин;
- при желании в глоссарий можно включить не только отельные слова и термины, но и целые фразы.

1.6. Методические рекомендации для занятий в интерактивной форме

В учебном процессе, помимо чтения лекций и аудиторных занятий, используются интерактивные формы. В сочетании с внеаудиторной работой это способствует формированию и развитию профессиональных навыков обучающихся.

Интерактивное обучение представляет собой способ познания, осуществляемый в формах совместной деятельности обучающихся, т.е. все участники образовательного процесса взаимодействуют друг с другом, совместно решают поставленные проблемы, моделируют ситуации, обмениваются информацией, оценивают действие коллег и свое собственное поведение, погружаются в реальную атмосферу делового сотрудничества по разрешению проблем.

В курсе изучаемой дисциплины «Программирование» в интерактивной форме часы используются в виде: обратной связи.

Тематика занятий с использованием интерактивных форм

№ п/п	Тема	Интерактивная форма	Часы, отводимые на интерактивные формы	
			Лекции	Лабораторные занятия
1.	Языки программирования	Обратная связь	1	-
2.	Неформальное введение в язык С++	Обратная связь	1	-
3.	Система программирования	Обратная связь	1	-
4.	Лексемы языка С++	Обратная связь	1	-
5.	Константы языка С++	Обратная связь	1	
6.	Типы языка С++	Обратная связь	1	
7.	Выражения языка С++	Обратная связь	1	
8.	Инструкции (операторы) языка С++	Обратная связь	1	
9.	Блочная структура программы на С++	Обратная связь	1	
10.	Подпрограммы языка С++ .	Обратная связь	1	
11.	Заголовочные файлы языка С++	Обратная связь	1	
12.	Использование файлов в программах на С++	Обратная связь	1	
13.	Реализация структур данных на языке С++	Обратная связь	2	
14.	Объектно-ориентированное программирование	Обратная связь	2	

№ п/п	Тема	Интерактивная форма	Часы, отводимые на интерактивные формы	
			Лекции	Лабораторные занятия
15.	Потоковые классы:	Обратная связь	2	
16.	Обработка исключительных ситуаций:	Обратная связь	2	
17.	Шаблоны функций и классов:	Обратная связь	2	
18.	Объектно-ориентированный анализ	Обратная связь	2	
ИТОГО			**24 часа**	

План лабораторных работ

3 семестр

Лабораторная работа № 1. "Массивы"

1. Постановка задачи

Все массивы должны быть динамическими.

2. Варианты

1. Написать функцию для обмена строк двумерного массива с ее помощью отсортировать массив по элементам третьего столбца.
2. Написать процедуру для суммирования матриц. С ее помощью сложить исходную матрицу и транспонированную (т. е. полученную поворотом исходной на 90).
3. Написать функцию для удаления строки из двумерного массива. Оставшиеся строки должны быть расположены плотно, недостающие элементы заменяются 0. С помощью разработанных функций исключить из массива строки с номерами от A до B.
4. Определить является ли матрица ортонормированной, т. е. такой, что скалярное произведение каждой пары различных строк равно 0, а скалярное произведение строки самой на себя равно 1.
5. Элемент матрицы является седловой точкой, если он является наименьшим в своей строке и наибольшим в своем столбце (или наоборот: наибольшим в своей строке и наименьшим в своем столбце).Для заданной матрицы определить все седловые точки.
6. Написать процедуру обмена столбца и строки двухмерного массива. С ее помощью поменять местами те строки и столбцы, первые элементы которых совпадают.
7. Написать функцию транспонирования квадратной матрицы (т.е. поворота исходной матрицы на 90). С ее помощью определить является ли заданная матрица симметрической. (Матрица называется симметрической, если транспонированная матрица равна исходной).
8. Написать функцию для вычисления суммы элементов квадратной матрицы, которые расположены ниже главной диагонали. С ее помощью найти максимальное значение такой суммы в n матрицах.
9. Написать функцию, проверяющую есть ли отрицательные элементы в указанной строке двумерного массива. Удалить из массива все строки с отрицательными элементами, удаленная строка заполняется 0 и переносится в конец массива.
10. Написать функцию, проверяющую по возрастанию или убыванию упорядочена указанная строка двумерного массива. Упорядочить по возрастанию все строки двумерного массива, которые неупорядочены по убыванию.
11. Написать функцию, для поиска максимального элемента в указанной строке двумерного массива. Сдвинуть в двумерном массиве все строки циклически вправо на количество элементов равное максимальному элементу в этой строке.
12. Определить можно ли в двумерном массиве найти такой столбец, который разбивает массив на два так, что сумма элементов в первом больше, чем сумма элементов во втором. Сам столбец в разбиваемые части не входит.

13. Вычислить произведение всех столбцов массива, у которых первый элемент больше элементов расположенных на главной и побочной диагонали.
14. Задан двумерный массив. Найти сумму элементов первого столбца без одного последнего элемента, сумму элементов второго столбца без двух последних, сумму элементов третьего столбца без трех последних и т. д. Последний столбец не обрабатывается. Среди найденных сумм найти максимальную.
15. Задан двумерный массив N x N. Разрешается произвольно переставлять элементы внутри любого столбца. Проверить, можно ли выполнив конечное количество перестановок в столбцах, расположить на побочной диагональ элементы так, чтобы он возрастали.
16. Задан двумерный массив N x M. Найти в нем подмассив 3 x 3, сумма элементов которого максимальна. N и M могут быть не кратны трем.
17. Задан двумерный массив N x N. Последовательно рассматриваются квадратные подмассивы, правый верхний элемент которых лежит на побочной диагонали. В каждом таком подмассиве находится максимальный элемент. Путем перестановок строк и столбцов (целиком) элемент надо переместить в правый верхний угол подмассива. Проверить получилась ли на побочной диагонали убывающая последовательность элементов.
18. Задана строка из N^2 цифр. Установить можно ли, разбив строку на подстроки длиной N, записать их в строки двумерного массива N x N по одной цифре в одном элементе так, чтобы они в первом столбце расположились в порядке возрастания.
19. Найти минимальный из неповторяющихся элементов двумерного массива.
20. Найти максимальный из повторяющихся элементов двумерного массива.
21. В двумерном массиве найти среднее арифметическое первого столбца и количество элементов в каждом из следующих столбцов, превышающих среднее арифметическое предыдущего столбца.
22. Задан одномерный массив состоящий из N целых чисел. Сформировать на его основе двумерный массив N x N так, чтобы сумма элементов в первом столбце была равна первому элементу одномерного массива, сумма элементов во втором столбце была равна второму элементу одномерного массива и т. д. Нули не использовать.
23. Определить сколько элементов двумерного массива больше любого элемента на главной диагонали.
24. Найти количество элементов в массиве, значение которых больше среднего значения всех элементов.

Лабораторная работа № 2. "Работа с файлами"

1. Условия выполнения задач

- Обучающемуся необходимо подготовить решение двух задач из нижележащего списка.
- Все исходные величины должны вводиться из файла (in.txt), а результат должен быть записан в результирующий файл (out.txt). Формат файла необходимо выбрать самостоятельно (например, текстовый фал in.txt, в котором необходимые значения перечислены через запятую). Исходный файл можно сформировать и заполнить вручную с помощью внешнего текстового редактора.

2. Варианты заданий

1. Даны два упорядоченных по возрастанию одномерных массива целых числе. Необходимо записать два исходных массива в третий так, чтобы он был тоже упорядоченным по возрастанию массивом.
2. Дан одномерный массив. Необходимо вывести на экран индексы максимального и минимального элементов данного массива
3. Дана одномерный массив целых числе. Необходимо подсчитать количество различных чисел в массиве. Например: массив 121332 содержит 3 различные цифры – это 1, 2 и 3

4. В файле дан набор чисел. Необходимо определить является ли каждое число простым. (Простое число – это число, которое делится нацело только на единицу и на самого себя). В Результирующий файл записать результат:
5 - yes
8 - no
11 - yes и т.д.
5. В массиве из 15 целых элементов необходимо поменять местами максимальный и минимальный элементы.
6. Даны два массива одной размерности - к1, к2, - целых чисел. Определите количество четных чисел на одинаковых местах во всех массивах.
7. В массиве из 20 целых чисел необходимо подсчитать количество положительных, отрицательных и нулевых элементов.
8. Дано целое число. Если число отрицательное, то необходимо вывести все четные числа, начиная со введенного до -2. Если число положительное, то необходимо вывести все нечетные числа, начиная с этого числа и заканчивая его квадратом.
9. Дан массив из N целых элементов и два целых числа A и B. Необходимо определить количество элементов, которые одновременно больше A и меньше B.
10. Дана строка символов. Необходимо удвоить все пробелы в строке.
11. Дан массив из N вещественных чисел и вещественное число R. Необходимо найти элемент массива, который максимально близок или равен R.
12. Даны две строки символов S и T. Необходимо определить сколько раз строка T входит в строку S и удалить первое и последнее вхождение T в S.
13. Дан массив из N вещественных чисел. Необходимо найти суммы максимальных и минимальных элементов.
14. Дан массив из N вещественных чисел. Необходимо найти сумму всех элементов массива и записать ее на место первого элемента массива.
15. Дан массив из N вещественных чисел. Если количество элементов массива четно, то в пару элементов, которые находятся посередине массива записать сумму элементов от начала до середины и сумму элементов от середины до конца массива соответственно. Если N – не четно, то записать в центральный элемент массива 0.
16. Среди данного массива символов, найти и вывести все имеющиеся пары стоящих рядом одинаковых символов
17. Дан массив из 40 целых чисел. Найти максимальное число в этом массиве.
18. Дано 100 вещественных чисел. Вычислить разность между максимальным и минимальным из них.
19. Дан массив, в который записаны элементы последовательности натуральных чисел. Количество этих элементов не известно, но известно, что последним элементом последовательности является 0. Необходимо определить порядковый номер наименьшего элемента этой последовательности. (примечание: массив может быть заполнен не полностью, поэтому говорится, что кол-во элементов не известно)
20. Дано целое $n > 0$ и последовательность из n любых вещественных чисел, среди которых есть хотя бы одно отрицательное число. Найти элемент наибольший среди отрицательных чисел этой последовательности.
21. Дано n вещественных чисел. Определить, образуют ли они знакочередующуюся последовательность. (например, знакочередующаяся последовательность: -1.2, 2.8, -3.7, 9.1, -4.2, 1.0)
22. Дан массив, в который записаны элементы последовательности натуральных чисел. Количество этих элементов не известно, но известно, что последним элементом последовательности является 0. Найти количество четных элементов этой последовательности.
23. Дана последовательность из n вещественных чисел. Найти наиболее длинную последовательность подряд идущих нулей в этой последовательности.

24. Дана последовательность из 40 целых чисел. Логической переменной присвоить значение True, если в последовательности нет нулевых элементов и при этом положительные элементы чередуются с отрицательными, и значение False иначе.
25. Даны последовательность из n целых чисел и целое число a. Найти порядковый номер первого вхождения a в последовательность (номер первого элемента равного a) или вывести сообщение, что элемента равного a в последовательности нет.
26. Дана последовательность из n вещественных чисел. Найти значение последнего отрицательного элемента.
27. Дан массив, в который записаны элементы последовательности натуральных чисел. Количество этих элементов не известно, но известно, что последним элементом последовательности является 0. Если на четном месте находится элемент кратный 3, то заменить этот элемент на его квадрат. Полученную последовательность вывести на экран.
28. Дана последовательность из m вещественных чисел. Найти количество элементов, которые больше своих соседей (левого: $a_n > a_{n-1}$ и правого: $a_n > a_{n+1}$).
29. Дана последовательность из 70 целых чисел. Определить сколько отрицательных чисел стоит в начале этой последовательности.
30. Дана последовательность, состоящая из 100 вещественных чисел. Определить является ли она возрастающей или убывающей.
31. Дана последовательность из 50 чисел. Найти их среднее арифметическое этих чисел.
32. Дан массив целых чисел. Необходимо подсчитать сумму элементов массива, которые отличны от последнего числа массива.
33. Дана последовательность из *n* вещественных чисел. Вычислить сумму тех элементов последовательности, номера которых совпадают со значениями элементов последовательности.
34. Дана последовательность из *n* вещественных чисел. Необходимо замены отрицательные элементы последовательности их квадратами, а из положительных извлечь квадратный корень. Подсчитать сумму элементов полученной последовательности.
35. Дан массив вещественных чисел. Если сумма первого и последнего элемента окажется положительной, то необходимо получить сумму элементов исходной последовательности, а если отрицательной, то необходимо получить их произведение.
36. Даны две последовательности из *n* вещественных чисел каждая X и Y. Необходимо сформировать третью последовательность по следующему правилу: первый элемент результирующей последовательности равен $max(x_1,y_1)$; второй - $max(x_2, y_2)$ третий- $max(x_3 y_3)$
37. Дана последовательность из *n* вещественных чисел. Необходимо сформировать две новые последовательности таким образом, чтобы в первой были записаны все положительные, а во второй все отрицательные элементы исходной последовательности.
38. Дана последовательность из *n* вещественных чисел. Сдвинуть все элементы последовательности циклически на *k* позиций влево. Например: 12345678 – на три позиции влево -> 45678123
39. Дана последовательность из ***n*** вещественных чисел. Переменной ***t*** присвоить значение **True**, если элементы последовательности упорядочены строго по возрастанию.
40. Дана последовательность из ***n*** вещественных чисел. Переменной *t* присвоить значение **True**, если в последовательности нет нулевых элементов и при этом положительные элементы чередуются с отрицательными.
41. Дана последовательность из *n* вещественных чисел. Все элементы первой половины последовательности с четными номерами, домножить на него максимальный элемент этой последовательности.
42. Дана последовательность из *n* вещественных чисел. Поменять в последовательности местами наибольший элемент и элемент с номером *m*.

43. Дана последовательность из *n* вещественных чисел. Найти номер первого вхождения данного числа в последовательность или вывести сообщение, что такого числа нет.
44. Дана последовательность из *n* вещественных чисел. Найти номер последнего вхождения данного числа в последовательность или вывести сообщение, что такого числа нет.
45. Дана последовательность из *n* натуральных чисел. Переменной *t* присвоить значение **True**, если среди элементов последовательности есть хотя бы одна пара чисел, таких что $a_i = a_{i-1}/2$.
46. Дана последовательность из n целых чисел. Удалить все элементы последовательности значения которых кратны *k* (делятся на k без остатка).
47. Дана последовательность *a* из 99 целых чисел. Получить новую последовательность, удалив из исходной все элементы со значением *max(a*1,..., *a*99).
48. Дана последовательность из *n* действительных чисел. Получить числа *b*1,...,*b*n, где *b*i - среднее арифметическое всех элементов исходной последовательности, кроме *i* - того элемента.
49. Дана строка символов. Сколько различных символов встречается в ней.
50. Дана последовательность из 20ти целых чисел. Первый по порядку элемент с наименьшим значением заменить целой частью среднего арифметического всех элементов исходной последовательности, остальные элементы оставить без изменения.

Лабораторная работа № 3. "Функции и массивы"

1. Постановка задачи

Используя функции, решить указанную в варианте задачу. Необходимые данные должны передаваться в функцию как параметр. Использование глобальных переменных запрещается.
Результат выполнения программы записать в файл out.txt

2. Варианты

1. Написать функцию для обмена строк двумерного массива с ее помощью отсортировать массив по элементам третьего столбца.
2. Написать процедуру для суммирования матриц. С ее помощью сложить исходную матрицу и транспонированную (т. е. полученную поворотом исходной на 90 °).
3. Написать функцию для удаления строки из двумерного массива. Оставшиеся строки должны быть расположены плотно, недостающие элементы заменяются 0. С помощью разработанных функций исключить из массива строки с номерами от A до B.
4. Определить является ли матрица ортонормированной, т. е. такой, что скалярное произведение каждой пары различных строк равно 0, а скалярное произведение строки самой на себя равно 1.
5. Элемент матрицы является седловой точкой, если он является наименьшим в своей строке и наибольшим в своем столбце (или наоборот: наибольшим в своей строке и наименьшим в своем столбце).Для заданной матрицы определить все седловые точки.
6. Написать процедуру обмена столбца и строки двухмерного массива. С ее помощью поменять местами те строки и столбцы, первые элементы которых совпадают.
7. Написать функцию транспонирования квадратной матрицы (т.е. поворота исходной матрицы на 90 °). С ее помощью определить является ли заданная матрица симметрической. (Матрица называется симметрической, если транспонированная матрица равна исходной).
8. Написать функцию для вычисления суммы элементов квадратной матрицы, которые расположены ниже главной диагонали. С ее помощью найти максимальное значение такой суммы в n матрицах.

9. Написать функцию, проверяющую есть ли отрицательные элементы в указанной строке двумерного массива. Удалить из массива все строки с отрицательными элементами, удаленная строка заполняется 0 и переносится в конец массива.
10. Написать функцию, проверяющую по возрастанию или убыванию упорядочена указанная строка двумерного массива. Упорядочить по возрастанию все строки двумерного массива, которые неупорядочены по убыванию.
11. Написать функцию, для поиска максимального элемента в указанной строке двумерного массива. Сдвинуть в двумерном массиве все строки циклически вправо на количество элементов равное максимальному элементу в этой строке.
12. Определить можно ли в двумерном массиве найти такой столбец, который разбивает массив на два так, что сумма элементов в первом больше, чем сумма элементов во втором. Сам столбец в разбиваемые части не входит.
13. Вычислить произведение всех столбцов массива, у которых первый элемент больше элементов расположенных на главной и побочной диагонали.
14. Задан двумерный массив. Найти сумму элементов первого столбца без одного последнего элемента, сумму элементов второго столбца без двух последних, сумму элементов третьего столбца без трех последних и т. д. Последний столбец не обрабатывается. Среди найденных сумм найти максимальную.
15. Задан двумерный массив N x N. Разрешается произвольно переставлять элементы внутри любого столбца. Проверить, можно ли выполнив конечное количество перестановок в столбцах, расположить на побочной диагональ элементы так, чтобы он возрастали.
16. Задан двумерный массив N x M. Найти в нем подмассив 3 x 3, сумма элементов которого максимальна. N и M могут быть не кратны трем.
17. Задан двумерный массив N x N. Последовательно рассматриваются квадратные подмассивы, правый верхний элемент которых лежит на побочной диагонали. В каждом таком подмассиве находится максимальный элемент. Путем перестановок строк и столбцов (целиком) элемент надо переместить в правый верхний угол подмассива. Проверить получилась ли на побочной диагонали убывающая последовательность элементов.
18. Задана строка из N^2 цифр. Установить можно ли, разбив строку на подстроки длиной N, записать их в строки двумерного массива N x N по одной цифре в одном элементе так, чтобы они в первом столбце расположились в порядке возрастания.
19. Найти минимальный из неповторяющихся элементов двумерного массива.
20. Найти максимальный из повторяющихся элементов двумерного массива.
21. В двумерном массиве найти среднее арифметическое первого столбца и количество элементов в каждом из следующих столбцов, превышающих среднее арифметическое предыдущего столбца.
22. Задан одномерный массив состоящий из N целых чисел. Сформировать на его основе двумерный массив N x N так, чтобы сумма элементов в первом столбце была равна первому элементу одномерного массива, сумма элементов во втором столбце была равна второму элементу одномерного массива и т. д. Нули не использовать.
23. Определить сколько элементов двумерного массива больше любого элемента на главной диагонали.
24. Найти количество элементов в массиве, значение которых больше среднего значения всех элементов.

Лабораторная работа № 4."Структуры"

1. Постановка задачи

Используя функции, решить указанную в варианте задачу. Необходимые данные должны передаваться в функцию как параметр. Использование глобальных переменных запрещается.

Результат выполнения программы записать в файл out.txt

2. Варианты

Вариант 1

1. Описать структуру с именем STUDENT, содержащую следующие поля:
• фамилия и инициалы;
• номер группы;
• успеваемость (массив из пяти элементов).
2. Написать программу, выполняющую следующие действия:
• ввод с клавиатуры данных в массив, состоящий из десяти структур типа STUDENT; записи должны быть упорядочены по возрастанию номера группы;
• вывод на дисплей фамилий и номеров групп для всех обучающихся, включенных в массив, если средний балл студента больше 4,0;
• если таких обучающихся нет, вывести соответствующее сообщение.

Вариант 2

1. Описать структуру с именем STUDENT, содержащую следующие поля:
• фамилия и инициалы;
• номер группы;
• успеваемость (массив из пяти элементов).
2. Написать программу, выполняющую следующие действия:
• ввод с клавиатуры данных в массив, состоящий из десяти структур типа STUDENT; записи должны быть упорядочены по возрастанию среднего балла;
• вывод на дисплей фамилий и номеров групп для всех обучающихся, имеющих оценки 4 и 5;
• если таких обучающихся нет, вывести соответствующее сообщение.

Вариант 3

1. Описать структуру с именем STUDENT, содержащую следующие поля:
• фамилия и инициалы;
• помер группы;
ш успеваемость (массив из пяти элементов).
2. Написать программу, выполняющую следующие действия:
• ввод с клавиатуры данных в массив, состоящий из десяти структур типа STUDENT; записи должны быть упорядочены по алфавиту;
ш вывод на дисплей фамилий и номеров групп для всех обучающихся, имеющих хотя бы одну оценку 2;
• если таких обучающихся нет, вывести соответствующее сообщение.

Вариант 4

1. Описать структуру с именем AEROFLOT, содержащую следующие поля:
• название пункта назначения рейса;
• номер рейса;
• тип самолета.
2. Написать программу, выполняющую следующие действия:
• ввод с клавиатуры данных в массив, состоящий из семи элементов типа AEROFLOT; записи должны быть упорядочены но возрастанию номера рейса;
• вывод на экран номеров рейсов и типов самолетов, вылетающих в пункт назначения, название которого совпало с названием, введенным с клавиатуры;
• если таких рейсов нет, выдать на дисплей соответствующее сообщение.

Вариант 5

1. Описать структуру с именем AEROFLOT, содержащую следующие поля:
• название пункта назначения рейса;
• номер рейса;
• тип самолета.
2. Написать программу, выполняющую следующие действия:

• ввод с клавиатуры данных в массив, состоящий из семи элементов типа AEROFLOT; записи должны быть размещены в алфавитном порядке по названиям пунктов назначения;
• вывод на экран пунктов назначения и номеров рейсов, обслуживаемых самолетом, тип которого введен с клавиатуры;
• если таких рейсов нет, выдать на дисплей соответствующее сообщение.

Вариант 6

1. Описать структуру с именем WORKER, содержащую следующие поля:
• фамилия и инициалы работника;
• название занимаемой должности;
• год поступления на работу.
2. Написать программу, выполняющую следующие действия:
• ввод с клавиатуры данных в массив, состоящий из десяти структур типа WORKER; записи должны быть размещены по алфавиту.
• вывод на дисплей фамилий работников, чей стаж работы в организации превышает значение, введенное с клавиатуры;
• если таких работников нет, вывести на дисплей соответствующее сообщение.

Вариант 7

1. Описать структуру с именем TRAIN, содержащую следующие поля:
• название пункта назначения;
• номер поезда;
• время отправления.
2. Написать программу, выполняющую следующие действия:
• ввод с клавиатуры данных в массив, состоящий из восьми элементов типа TRAIN; записи должны быть размещены в алфавитном порядке по названиям пунктов назначения;
• вывод на экран информации о поездах, отправляющихся после введеного с клавиатуры времени;
• если таких поездов нет, выдать на дисплей соответствующее сообщение.

Вариант 8

1. Описать структуру с именем TRAIN, содержащую следующие поля:
• название пункта назначения;
• номер поезда;
• время отправления.
2. Написать программу, выполняющую следующие действия:
• ввод с клавиатуры данных в массив, состоящий из шести элементов типа TRAIN; записи должны быть упорядочены по времени отправления поезда;
• вывод на экран информации о поездах, направляющихся в пункт, название которого введено с клавиатуры;
• если таких поездов нет, выдать на дисплей соответствующее сообщение.

Вариант 9

1. Описать структуру с именем TRAIN, содержащую следующие поля:
• название пункта назначения;
• номер поезда;
• время отправления.
2. Написать программу, выполняющую следующие действия:
• ввод с клавиатуры данных в массив, состоящий из восьми элементов типа TRAIN; записи должны быть упорядочены по номерам поездов;
• вывод на экран информации о поезде, номер которого введен с клавиатуры;
• если таких поездов нет, выдать на дисплей соответствующее сообщение.

Вариант 10

1. Описать структуру с именем MARSH, содержащую следующие поля:
• название начального пункта маршрута;

• название конечного пункта маршрута;
• номер маршрута.
2. Написать программу, выполняющую следующие действия:
• ввод с клавиатуры данных в массив, состоящий из восьми элементов типа MARSH; записи должны быть упорядочены по номерам маршрутов;
• вывод на экран информации о маршруте, номер которого введен с клавиатуры;
• если таких маршрутов нет, выдать на дисплей соответствующее сообщение.

Вариант 11

1. Описать структуру с именем MARSH, содержащую следующие поля:
• название начального пункта маршрута;
• название конечного пункта маршрута;
• номер маршрута.
2. Написать программу, выполняющую следующие действия:
• ввод с клавиатуры данных в массив, состоящий из восьми элементов типа MARSH; записи должны быть упорядочены по номерам маршрутов;
• вывод на экран информации о маршрутах, которые начинаются или кончаются в пункте, название которого введено с клавиатуры;
• если таких маршрутов нет, выдать на дисплей соответствующее сообщение.

Вариант 12

1. Описать структуру с именем NOTE, содержащую следующие поля:
• фамилия, имя;
• номер телефона;
• день рождения (массив из трех чисел).
2. Написать программу, выполняющую следующие действия:
• ввод с клавиатуры данных в массив, состоящий из восьми элементов типа NOTE; записи должны быть упорядочены но датам дней рождения;
• вывод на экран информации о человеке, номер телефона которого введен с клавиатуры;
• если такого нет, выдать на дисплей соответствующее сообщение.

Вариант 13

1. Описать структуру с именем NOTE, содержащую следующие поля:
• фамилия, имя;
• номер телефона;
• день рождения (массив из трех чисел).
2. Написать программу, выполняющую следующие действия:
• ввод с клавиатуры данных в массив, состоящий из восьми элементов типа NOTE; записи должны быть размещены но алфавиту;
• вывод на экран информации о людях, чьи дни рождения приходятся на месяц, значение которого введено с клавиатуры;
• если таких нет, выдать на дисплей соответствующее сообщение.

Вариант 14

1. Описать структуру с именем NOTE, содержащую следующие поля:
• фамилия, имя;
• номер телефона;
• день рождения (массив из трех чисел).
2. Написать программу, выполняющую следующие действия:
• ввод с клавиатуры данных в массив, состоящий из восьми элементов типа NOTE; записи должны быть упорядочены по трем первым цифрам номера телефона;
• вывод на экран информации о человеке, чья фамилия введена с клавиатуры;
• если такого нет, выдать на дисплей соответствующее сообщение.

Вариант 15

1. Описать структуру с именем ZNAK, содержащую следующие поля:
• фамилия, имя;

• знак Зодиака;
• день рождения (массив из трех чисел).
2. Написать программу, выполняющую следующие действия:
• ввод с клавиатуры данных в массив, состоящий из восьми элементов типа ZNAK; записи должны быть упорядочены по датам дней рождения;
• вывод на экран информации о человеке, чья фамилия введена с клавиатуры;
• если такого нет, выдать на дисплей соответствующее сообщение.

Вариант 16

1. Описать структуру с именем ZNAK, содержащую следующие поля:
• фамилия, имя;
• знак Зодиака;
• день рождения (массив из трех чисел).
2. Написать программу, выполняющую следующие действия:
• ввод с клавиатуры данных в массив, состоящий из восьми элементов типа ZNAK; записи должны быть упорядочены по датам дней рождения;
• вывод на экран информации о людях, родившихся под знаком, наименование которого введено с клавиатуры;
• если таких нет, выдать на дисплей соответствующее сообщение.

Вариант 17

1. Описать структуру с именем ZNAK, содержащую следующие поля:
• фамилия, имя;
• знак Зодиака;
• день рождения (массив из трех чисел).
2. Написать программу, выполняющую следующие действия:
• ввод с клавиатуры данных в массив, состоящий из восьми элементов типа ZNAK; записи должны быть упорядочены по знакам Зодиака;
• вывод на экран информации о людях, родившихся в месяц, значение которого введено с клавиатуры;
• если таких нет, выдать на дисплей соответствующее сообщение.

Вариант 18

1. Описать структуру с именем PRICE, содержащую следующие поля:
• название товара;
• название магазина, в котором продается товар;
• стоимость товара в руб.
2. Написать программу, выполняющую следующие действия:
• ввод с клавиатуры данных в массив, состоящий из восьми элементов типа PRICE; записи должны быть размещены в алфавитном порядке по названиям товаров;
• вывод на экран информации о товаре, название которого введено с клавиатуры;
• если таких товаров нет, выдать на дисплей соответствующее сообщение.

Вариант 19

1. Описать структуру с именем PRICE, содержащую следующие поля:
• название товара;
• название магазина, в котором продается товар;
• стоимость товара в руб.
2. Написать программу, выполняющую следующие действия:
• ввод с клавиатуры данных в массив, состоящий из восьми элементов типа PRICE; записи должны быть размещены в алфавитном порядке по названиям магазинов;
• вывод на экран информации о товарах, продающихся в магазине, название которого введено с клавиатуры;
• если такого магазина нет, выдать на дисплей соответствующее сообщение.

Вариант 20

1. Описать структуру с именем ORDER, содержащую следующие поля:

• расчетный счет плательщика;
• расчетный счет получателя;
• перечисляемая сумма в руб.
2. Написать программу, выполняющую следующие действия:
• ввод с клавиатуры данных в массив, состоящий из восьми элементов типа ORDER; записи должны быть размещены в алфавитном порядке по расчетным счетам плательщиков;
• вывод па экран информации о сумме, снятой с расчетного счета плательщика, введенного с клавиатуры;
• если такого расчетного счета нет, выдать на дисплей соответствующее сообщение.

4 семестр

Лабораторная работа № 1. Классы и объекты в С++

Постановка задачи

Написать программу, в которой создаются и разрушаются объекты, определенного пользователем класса. Выполнить исследование вызовов конструкторов и деструкторов.

Порядок выполнения работы

1. Определить пользовательский класс в соответствии с вариантом задания (смотри приложение).
2. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.
3. Определить в классе деструктор.
4. Определить в классе компоненты-функции для просмотра и установки полей данных.
5. Определить указатель на экземпляр класса.
6. Написать демонстрационную программу, в которой создаются и разрушаются объекты пользовательского класса и каждый вызов конструктора и деструктора сопровождается выдачей соответствующего сообщения (какой объект какой конструктор или деструктор вызвал).
7. Показать в программе использование указателя на объект.
8. Лабораторная должна состоять из трёх файлов:
- заголовочный h-файл с определением класса,
- cpp-файл с реализацией класса,
- cpp-файл демонстрационной программы.

Для предотвращения многократного включения файла-заголовка следует использовать директивы препроцессора

```
#ifndef STUDENTH
#define STUDENTH
// модуль STUDENT.H
...
#endif
```

Варианты заданий.

Описания членов - данных пользовательских классов

1. СТУДЕНТ
имя – char*
курс – int
пол – int(bool)

4. ИЗДЕЛИЕ
имя – char*
шифр – char*
количество – int

7. АДРЕС
имя – char*
улица – char*
номер дома – int

10. ЦЕХ
имя – char*
начальник – char*
количество
работающих – int

13. СТРАНА
имя – char*

форма
правления – char*
площадь – float

2. СЛУЖАЩИЙ
имя – char*
возраст – int
рабочий стаж – int

5. БИБЛИОТЕКА
имя – char*
автор – char*
стоимость – float

8. ТОВАР
имя – char*
количество – int
стоимость – float

11. ПЕРСОНА
имя – char*
возраст – int
пол – int(bool)

14. ЖИВОТНОЕ
имя – char*
класс – char*
средний вес – int

3. КАДРЫ
имя – char*
номер цеха – int
разряд – int
6. ЭКЗАМЕН
имя студента – char*
дата – int
оценка – int

9. КВИТАНЦИЯ
номер – int
дата – int
сумма – float

12. АВТОМОБИЛЬ
марка – char*
мощность – int
стоимось – float
15. КОРАБЛЬ
имя – char*
водоизмещение – int
тип – char*

Лабораторная работа № 2. Перегрузка операций

Основное содержание работы.

Определить и реализовать класс. Определить и реализовать операции над данными этого класса. Написать и выполнить программу полного тестирования этого класса.

Порядок выполнения работы.

1. Выбрать класс в соответствии с вариантом.
2. Определить и реализовать в классе конструкторы, деструктор, функции Input (ввод с клавиатуры) и Print (вывод на экран), перегрузить операцию присваивания.
3. Написать программу тестирования класса и выполнить тестирование.
4. Дополнить определение класса заданными перегруженными операциями (в соответствии с вариантом).
5. Реализовать эти операции. Выполнить тестирование.

Методические указания.

1.Класс реализовать как динамический массив. Для этого определение класса должно иметь следующие поля:

– указатель на начало массива;

– максимальный размер массива;

– текущий размер массива.

2. Конструкторы класса размещают массив в памяти и устанавливают его максимальный и текущий размер. Для задания максимального массива использовать константу, определяемую вне класса.

3. Чтобы у вас не возникало проблем, аккуратно работайте с константными объектами. Например:

0* конструктор копирования следует определить так:

```
MyClass (const MyClass& ob);
```

1* операцию присваивания перегрузить так:

```
MyClass& operator = (const MyClass& ob);
```

4. Для удобства реализации операций-функций реализовать в классе **private(protected)**-функции, работающие непосредственно с реализацией класса. Например, для класса **множество** это могут быть следующие функции:

– включить элемент в множество;

– найти элемент и возвратить его индекс;

– удалить элемент;

– определить, принадлежит ли элемент множеству.

Указанные функции используются в реализации общедоступных функций-операций (operator).

Варианты заданий.

1. Множество с элементами типа **char.** Дополнительно перегрузить следующие операции:

+ – добавить элемент в множество(типа char + set);

+ – объединение множеств;

= = – проверка множеств на равенство.

2. Множество с элементами типа **char.** Дополнительно перегрузить следующие операции:

- – удалить элемент из множества (типа set-char);

* – пересечение множеств;

< – сравнение множеств.

3. Множество с элементами типа **char.** Дополнительно перегрузить следующие операции:

- – удалить элемент из множества (типа set-char);
> – проверка на подмножество;
!= – проверка множеств на неравенство.

4. Множество с элементами типа **char.** Дополнительно перегрузить следующие операции:
+ – добавить элемент в множество (типа set+char);
* – пересечение множеств;
= = – проверка множеств на равенство.

5. Множество с элементами типа **char.** Дополнительно перегрузить следующие операции:
+ – добавить элемент в множество(типа char + set);
+ – объединение множеств;
<= – сравнение множеств .

6. Множество с элементами типа **char.** Дополнительно перегрузить следующие операции:
> – проверка на принадлежность(char > set);
* – пересечение множеств;
< – проверка на подмножество.

7. Однонаправленный список с элементами типа **int.** Дополнительно перегрузить следующие операции:
+ – объединить списки (list+list);
-- – удалить элемент из начала (--list);
= = – проверка на равенство.

8. Однонаправленный список с элементами типа **int.** Дополнительно перегрузить следующие операции:
+ – добавить элемент в начало(int+list);
-- – удалить элемент из начала(–list);
= = – проверка на равенство.

9. Однонаправленный список с элементами типа **int.** Дополнительно перегрузить следующие операции:
+ – добавить элемент в конец (list+int);
-- – удалить элемент из конца (list--);
!= – проверка на неравенство.

10. Однонаправленный список с элементами типа **int.** Дополнительно перегрузить следующие операции:
[] – доступ к элементу в заданной позиции, например:
int i,c;
list L;
c=L[i];
+ – объединить два списка;
= = – проверка на равенство.

11. Однонаправленный список с элементами типа **char.** Дополнительно перегрузить следующие операции:
[] – доступ к элементу в заданной позиции, например:

int i; char c;
list L;
c=L[i];
+ – объединить два списка;
!= – проверка на неравенство.

12. Стек. Дополнительно перегрузить следующие операции:
+ – добавить элемент в стек;
– – извлечь элемент из стека;
bool()– проверка, пустой ли стек.

13. Очередь. Дополнительно перегрузить следующие операции:
+ – добавить элемент;
– – извлечь элемент;
bool() – проверка, пустая ли очередь.

14. Одномерный массив (вектор) вещественных чисел. Дополнительно перегрузить следующие операции:

+ – сложение векторов (a[i]+b[i] для всех i);
[] – доступ по индексу;
+ – добавить число к вектору (double+vector).

15. Одномерный массив (вектор) вещественных чисел. Дополнительно перегрузить следующие операции:
- – вычитание векторов (a[i]-b[i] для всех i);
[] – доступ по индексу;
- – вычесть из вектора число (vector-double).

16. Одномерный массив (вектор) вещественных чисел. Дополнительно перегрузить следующие операции:
* – умножение векторов (a[i]*b[i] для всех i);
[] – доступ по индексу;
* – умножить вектор на число (vector*double).

17. Одномерный массив (вектор) вещественных чисел. Дополнительно перегрузить следующие операции:
= – присвоить всем элементам вектора значение (vector=double);
[] – доступ по индексу;
= = – проверка на равенство;
!= – проверка на неравенство;

18. Двухмерный массив (матрица) вещественных чисел. Дополнительно перегрузить следующие операции:
* – умножение матриц;
* – умножение матрицы на число;
* – умножение числа на матрицу.

19. Двухмерный массив (матрица) вещественных чисел. Дополнительно перегрузить следующие операции:
- – разность матриц;
- – вычесть из матрицы число;
= = – проверка матриц на равенство.

20. Двухмерный массив (матрица) вещественных чисел. Дополнительно перегрузить следующие операции:
= – присвоить всем элементам матрицы значение (matr=double);
+ – сложение матриц;
+ – сложить матрицу с числом (matr+double).

21. Двухмерный массив (матрица) вещественных чисел. Дополнительно перегрузить следующие операции:
= = – проверка матриц на равенство;
++ – транспонировать матрицу.
-- – вычесть из всех элементов матрицы единицу.

Лабораторная работа № 3. Перегрузка операций

1. Постановка задачи

Написать программу, в которой создаются и разрушаются объекты, определенного пользователем класса. В данном классе реализована перегрузка операторов, указанных в задании. Все исходные величины должны вводиться из файла (in.txt), а результат должен быть записан в результирующий файл (out.txt). Формат файла необходимо выбрать самостоятельно (например, текстовый фал in.txt, в котором необходимые значения перечислены через запятую). Исходный файл можно сформировать и заполнить вручную с помощью внешнего текстового редактора.

2. Порядок выполнения работы

1. Определить пользовательский класс в соответствии с вариантом задания (смотри приложение).
2. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.
3. Определить в классе деструктор.
4. Определить в классе компоненты-функции для просмотра и установки полей данных.
5. В демонстрационной программе показать использование перегруженных операторов. Каждый вызов перегруженного оператора должен сопровождаться выдачей соответствующего сообщения на экран.
6. В демонстрационной программе должны создаваться и разрушаться объекты пользовательского класса и каждый вызов конструктора и деструктора сопровождается выдачей соответствующего сообщения (какой объект какой конструктор или деструктор вызвал).
7. Лабораторная работа должна состоять из трёх файлов:
   2* заголовочный h-файл с определением класса,
   3* cpp-файл с реализацией класса,
   4* cpp-файл демонстрационной программы.

3. Варианты заданий

1. Определить класс, в котором описать указатель на целое число. Реализовать в этом классе конструктор и деструктор, который, соответственно, выделяет и освобождает память для указателя. Перегрузить для данного класса операторы потокового ввода вывода (<<, >>) и X++, >, <. Продемонстрировать работу объектов класса.

2. Определить класс, в котором описать массив элементов. Реализовать в этом классе конструктор, который заполняет массив случайными значениями. Перегрузить для данного класса операторы потокового ввода вывода (<<, >>) и операторы ++X, +, -.Продемонстрировать работу объектов класса.

3. Определить класс, в котором описать С-строку из 25 элементов. Добавить в класс конструктор по умолчанию и конструктор копирования. Перегрузить для данного класса операторы потокового ввода вывода (<<, >>) и операторы ==, +, -. Продемонстрировать работу объектов класса.

4. Определить класс, в котором описать целое беззнаковое поле. Для объектов данного класса перегрузить операцию сложения и сравнения. Также перегрузить для данного класса операторы потокового ввода вывода (<<, >>) и операторы X++, ++X, -. Продемонстрировать работу объектов класса.

5. Определить класс, в котором описать два целых числа. Добавить в класс конструктор и деструктор. Конструктор присваивает полям класса случайные числа от 0 до 10. Деструктор обнуляет эти числа и выводит сообщение на экран о том, что объект «умер». Перегрузить для данного класса операторы потокового ввода вывода (<<, >>) и операторы ==, X--, <. Продемонстрировать работу объектов класса.

6. Определить класс, в котором описать С-строку из 100 символов. Добавить в класс конструктор по умолчанию, который должен инициализировать строку случайными значениями. Перегрузить для данного класса операторы потокового ввода вывода (<<, >>) и операторы ==, <, >

7. Определить класс, в котором описать два поля целого типа. Конструктор по умолчанию должен инициализировать поля случайными значениями. Определить дружественный (friend) оператор, который позволяет складывать и вычитать целые числа и объекты данного класса. Также перегрузить для данного класса операторы потокового ввода вывода (<<, >>).

8. Определить класс, в котором описаны поля, представляющие дату (день, месяц, год). Реализовать в классе конструктор по умолчанию, который инициализирует дату случайными значениями, а также конструктор, который запрашивает дату с клавиатуры. Перегрузить для данного класса операцию +.

9. Создать класс, состоящий из двух целых чисел, конструктора и деструктора. Конструктор присваивает этим числам случайное число от -10 до 10. Деструктор обнуляет эти числа. Для экземпляров этого класса перегрузить операцию сложения и ввода и вывода в поток (<<, >>)

10. Создать класс, состоящий из строки длинной 100 символов и двух функций. Первая функция – подсчет количества пробелов в строке, вторая – заменяет в строке символы «а» на «о». Реализовать в классе перегруженные операторы ввода-вывода в поток (<<, >>).

11. Создать класс, состоящий из строки длинной 50 символов, конструктора и деструктора. Конструктор присваивает строке произвольное значение. Деструктор обнуляет строку. Перегрузить операцию сложения двух экземпляров этого класса, а также операции ввода-вывода в поток (<<, >>).

12. Создать класс, состоящий из двух полей целого типа и двух функций. Первая функция – заменяет меньшее из чисел на их сумму, вторая – заменяет большее на их разность. Перегрузить операцию сложения двух экземпляров этого класса, а также операции ввода-вывода в поток (<<, >>).

13. Создать класс, состоящий из двух полей целого типа, функции и перегруженного оператора +. Функция – вычисляет разницу этих числе, Оператор возвращает объект данного класса, где первое число равно сумме первых чисел складываемых объектов,

второе число равно сумме вторых чисел складываемых объектов. Перегрузить операцию сложения двух экземпляров этого класса, а также операции ввода-вывода в поток (<<, >>).

14. Написать перегруженный оператор сравнения (==) и оператор вывода (<<) для класса, состоящего из двух чисел. Конструктор по умолчанию в классе инициализирует эти два числа, значениями, вводимыми с клавиатуры. Также перегрузить оператор сложения для экземпляров этого класса.

15. Создать класс, состоящий из двух массивов целых чисел и двух функций. Первая функция – выводит минимальный из максимальных элементов массивов, вторая – инициализирует массивы случайными числами. Для данного класса необходимо перегрузить оператор присваивания (=) и оператор сравнения (==).

16. Создать класс «римские числа» (I, II, III, IV, V, … X). Конструктор по умолчанию определяет случайное значение. Для экземпляров этого класса определит перегруженные операторы ввода-вывода (<<, >>), а также арифметические операции.

17. Создать класс «матрица». Конструктор заполняет матрицу последовательными значениями от -10 с шагом 1.5. Реализовать в классе функции: транспонирование, поиск минимального элемента, поиск максимального элемента, умножение на число. Перегрузить оператор вывода в поток (<<).

18. Создать класс матрица. Конструктор заполняет матрицу случайными значениями. Реализовать в классе функции: поиск суммы, поиск количества отрицательных элементов, умножение на число. Перегрузить оператор вывода в поток (<<)

19. Создать класс вектор. Конструктор по умолчанию создает случайны единичный вектор. Определить для класса перегруженные функции ввода и вывода (<<, >>) и сложения векторов, а также создать функции «поворот вектора на угол А», «отображение относительно заданной оси координат»

20. Создать класс и описать в нем указанные поля. В классе определить конструктор, в котором инициализировать поля случайными, но разумными данными. В классе определить деструктор, который будет удалять из памяти все динамически созданные в класс переменные. Также перегрузить операторы сравнения (==), присваивания (=) и вывода в поток (<<): СТУДЕНТ: имя – char*;курс – int; пол – int(bool)

21. Создать класс и описать в нем указанные поля. В классе определить конструктор, в котором инициализировать поля случайными, но разумными данными. В классе определить деструктор, который будет удалять из памяти все динамически созданные в класс переменные. Также перегрузить операторы сравнения (==), присваивания (=) и вывода в поток (<<): ИЗДЕЛИЕ: имя – char*; шифр – char*; количество – int

22. Создать класс и описать в нем указанные поля. В классе определить конструктор, в котором инициализировать поля случайными, но разумными данными. В классе определить деструктор, который будет удалять из памяти все динамически созданные в класс переменные. Также перегрузить операторы сравнения (==), присваивания (=) и вывода в поток (<<): АДРЕС: имя – char*; улица – char*; номер дома – int

23. Создать класс и описать в нем указанные поля. В классе определить конструктор, в котором инициализировать поля случайными, но разумными данными. В классе определить деструктор, который будет удалять из памяти все динамически созданные в класс переменные. Также перегрузить операторы сравнения (==), присваивания (=) и вывода в поток (<<): ЦЕХ: имя – char*; начальник – char*; количество работающих – int

24. Создать класс и описать в нем указанные поля. В классе определить конструктор, в котором инициализировать поля случайными, но разумными данными. В классе определить деструктор, который будет удалять из памяти все динамически созданные в

класс переменные. Также перегрузить операторы сравнения (==), присваивания (=) и вывода в поток (<<): СТРАНА: имя – char*; форма правления – char*; площадь – float

25. Создать класс и описать в нем указанные поля. В классе определить конструктор, в котором инициализировать поля случайными, но разумными данными. В классе определить деструктор, который будет удалять из памяти все динамически созданные в класс переменные. Также перегрузить операторы сравнения (==), присваивания (=) и вывода в поток (<<): СЛУЖАЩИЙ: имя – char*; возраст – int; рабочий стаж – int

26. Создать класс и описать в нем указанные поля. В классе определить конструктор, в котором инициализировать поля случайными, но разумными данными. В классе определить деструктор, который будет удалять из памяти все динамически созданные в класс переменные. Также перегрузить операторы сравнения (==), присваивания (=) и вывода в поток (<<): БИБЛИОТЕКА: имя – char*; автор – char*; стоимость – float

27. Создать класс и описать в нем указанные поля. В классе определить конструктор, в котором инициализировать поля случайными, но разумными данными. В классе определить деструктор, который будет удалять из памяти все динамически созданные в класс переменные. Также перегрузить операторы сравнения (==), присваивания (=) и вывода в поток (<<): ТОВАР: имя – char*; количество – int; стоимость – float

28. Создать класс и описать в нем указанные поля. В классе определить конструктор, в котором инициализировать поля случайными, но разумными данными. В классе определить деструктор, который будет удалять из памяти все динамически созданные в класс переменные. Также перегрузить операторы сравнения (==), присваивания (=) и вывода в поток (<<): ПЕРСОНА: имя – char*; возраст – int; пол – int(bool)

29. Создать класс и описать в нем указанные поля. В классе определить конструктор, в котором инициализировать поля случайными, но разумными данными. В классе определить деструктор, который будет удалять из памяти все динамически созданные в класс переменные. Также перегрузить операторы сравнения (==), присваивания (=) и вывода в поток (<<): ЖИВОТНОЕ: имя – char*; класс – char*; средний вес – int

30. Создать класс и описать в нем указанные поля. В классе определить конструктор, в котором инициализировать поля случайными, но разумными данными. В классе определить деструктор, который будет удалять из памяти все динамически созданные в класс переменные. Также перегрузить операторы сравнения (==), присваивания (=) и вывода в поток (<<): КАДРЫ: имя – char*; номер цеха – int; разряд – int

31. Создать класс и описать в нем указанные поля. В классе определить конструктор, в котором инициализировать поля случайными, но разумными данными. В классе определить деструктор, который будет удалять из памяти все динамически созданные в класс переменные. Также перегрузить операторы сравнения (==), присваивания (=) и вывода в поток (<<): ЭКЗАМЕН: имя студента – char*; дата – int; оценка – int

32. Создать класс и описать в нем указанные поля. В классе определить конструктор, в котором инициализировать поля случайными, но разумными данными. В классе определить деструктор, который будет удалять из памяти все динамически созданные в класс переменные. Также перегрузить операторы сравнения (==), присваивания (=) и вывода в поток (<<): КВИТАНЦИЯ: номер – int; дата – int; сумма – float

33. Создать класс и описать в нем указанные поля. В классе определить конструктор, в котором инициализировать поля случайными, но разумными данными. В классе определить деструктор, который будет удалять из памяти все динамически созданные в класс переменные. Также перегрузить операторы сравнения (==), присваивания (=) и вывода в поток (<<): АВТОМОБИЛЬ: марка – char*; мощность – int; стоимость – float

34. Создать класс и описать в нем указанные поля. В классе определить конструктор, в котором инициализировать поля случайными, но разумными данными. В классе определить деструктор, который будет удалять из памяти все динамически созданные в класс переменные. Также перегрузить операторы сравнения (==), присваивания (=) и вывода в поток (<<): КОРАБЛЬ: имя – char*; водоизмещение – int; тип – char*

35. Создать класс и описать в нем указанные поля. В классе определить конструктор, в котором инициализировать поля случайными, но разумными данными. В классе определить деструктор, который будет удалять из памяти все динамически созданные в класс переменные. Также перегрузить операторы сравнения (==), присваивания (=) и вывода в поток (<<): ПРОЦЕССОР: название – char*; частота – float; цена – float

36. Создать класс и описать в нем указанные поля. В классе определить конструктор, в котором инициализировать поля случайными, но разумными данными. В классе определить деструктор, который будет удалять из памяти все динамически созданные в класс переменные. Также перегрузить операторы сравнения (==), присваивания (=) и вывода в поток (<<): ПРЕЗИДЕНТ: имя – char*; страна – char*; поддержка населения % – float

37. Создать класс и описать в нем указанные поля. В классе определить конструктор, в котором инициализировать поля случайными, но разумными данными. В классе определить деструктор, который будет удалять из памяти все динамически созданные в класс переменные. Также перегрузить операторы сравнения (==), присваивания (=) и вывода в поток (<<): ВЫБОРЫ: учреждение – char*; кандидаты – char*; расходы – int

38. Создать класс и описать в нем указанные поля. В классе определить конструктор, в котором инициализировать поля случайными, но разумными данными. В классе определить деструктор, который будет удалять из памяти все динамически созданные в класс переменные. Также перегрузить операторы сравнения (==), присваивания (=) и вывода в поток (<<): УЧЕНЫЙ: имя – char*; наука – char*; годы открытий – short[];

39. Создать класс и описать в нем указанные поля. В классе определить конструктор, в котором инициализировать поля случайными, но разумными данными. В классе определить деструктор, который будет удалять из памяти все динамически созданные в класс переменные. Также перегрузить операторы сравнения (==), присваивания (=) и вывода в поток (<<):СТАТЬЯ: название – char*; количество строк – short; автор – char*

40. Создать класс и описать в нем указанные поля. В классе определить конструктор, в котором инициализировать поля случайными, но разумными данными. В классе определить деструктор, который будет удалять из памяти все динамически созданные в класс переменные. Также перегрузить операторы сравнения (==), присваивания (=) и вывода в поток (<<):ПРАЗДНИК: название – char*; дата – date; меню – char*

41. Создать класс и описать в нем указанные поля. В классе определить конструктор, в котором инициализировать поля случайными, но разумными данными. В классе определить деструктор, который будет удалять из памяти все динамически созданные в класс переменные. Также перегрузить операторы сравнения (==), присваивания (=) и вывода в поток (<<):КОНТРОЛЬНАЯ: предмет – char*; сложность% -short; дата – date

42. Создать класс и описать в нем указанные поля. В классе определить конструктор, в котором инициализировать поля случайными, но разумными данными. В классе определить деструктор, который будет удалять из памяти все динамически созданные в класс переменные. Также перегрузить операторы сравнения (==), присваивания (=) и вывода в поток (<<): РАБОЧИЙ: имя – char*; возраст – int; рабочий стаж – int

Лабораторная работа № 4. Наследование и виртуальные функции

Основное содержание работы.
Написать программу, в которой создается иерархия классов. Включить полиморфные объекты в связанный список, используя статические компоненты класса. Показать использование виртуальных функций.

Порядок выполнения работы.

1. Определить иерархию классов (в соответствии с вариантом).
2. Определить в классе статическую компоненту - указатель на начало связанного списка объектов и статическую функцию для просмотра списка.
3. Реализовать классы.
4. Написать демонстрационную программу, в которой создаются объекты различных классов и помещаются в список, после чего список просматривается.

Методические указания.

1. Для определения иерархии классов связать отношением наследования классы, приведенные в задании. Из перечисленных классов выбрать один, который будет стоять во главе иерархии. Это абстрактный класс.
2. Определить в классах все необходимые конструкторы и деструктор.
3. Компонентные данные класса специфицировать как **protected**.
4. Пример определения статических компонентов:

static person* begin; // указатель на начало списка
static void print(void); // просмотр списка

5. Статическую компоненту-данное инициализировать вне определения класса, в глобальной области.
6. Для добавления объекта в список предусмотреть метод класса, т.е. объект сам добавляет себя в список. Например, a.Add() – объект **a** добавляет себя в список.
7. Включение объекта в список можно выполнять при создании объекта, т.е. поместить операторы включения в конструктор. В случае иерархии классов, включение объекта в список должен выполнять **только** конструктор базового класса. Вы должны продемонстрировать оба этих способа.
8. Список просматривать путем вызова виртуального метода **Show** каждого объекта.
9. Статический метод просмотра списка вызывать не через объект, а через класс.
10. Определение классов, их реализацию, демонстрационную программу поместить в отдельные файлы.

Варианты заданий .

Перечень классов:

1. студент, преподаватель, персона, завкафедрой;
2. служащий, персона, рабочий, инженер;
3. рабочий, кадры, инженер, администрация;
4. деталь, механизм, изделие, узел;
5. организация, страховая компания, судостроительная компания, завод;
6. журнал, книга, печатное издание, учебник;
7. тест, экзамен, выпускной экзамен, испытание;
8. место, область, город, мегаполис;
9. игрушка, продукт, товар, молочный продукт;
10. квитанция, накладная, документ, чек;
11. автомобиль, поезд, транспортное средство, экспресс;
12. двигатель, двигатель внутреннего сгорания, дизель, турбореактивный двигатель;
13. республика, монархия, королевство, государство;
14. млекопитающие, парнокопытные, птицы, животное;
15. корабль, пароход, парусник, корвет.

16. яблоко, фрукт, банан, персик
17. доктор, кандидат, звание, доцент
18. ботинки, обувь, тапочки, сапоги

Лабораторная работа № 5. Иерархия объектов и группа. Итераторы.

Основное содержание работы.

Получить практические навыки создания объектов-групп и использования методов-итераторов.

Основные теоретические сведения.

а. Группа.

Группа – это объект, в который включены другие объекты. Объекты, входящие в группу, называются элементами группы. Элементы группы, в свою очередь, могут быть группой.
Примеры групп:
1. Окно в интерактивной программе, которое владеет такими элементами, как поля ввода и редактирования данных, кнопки, списки выбора, диалоговые окна и т.д. Примерами таких окон являются объекты классов, порожденных от абстрактного класса TGroup(TDeskTop, TWindow, TDialog) в иерархии классов библиотеки Turbo Vision, и объекты классов, порожденных от TWindowObject в иерархии классов библиотеки OWL.
2. Агрегат, состоящий из более мелких узлов.
3. Огород, состоящий из растений, системы полива и плана выращивания.
4. Некая организационная структура (например, ФАКУЛЬТЕТ, КАФЕДРА, СТУДЕНЧЕСКАЯ ГРУППА).

Мы понимаем группу как класс, который не только хранит объекты других классов, но и обладает собственными свойствами, не вытекающими из свойств его элементов.
Группа дает второй вид иерархии (первый вид – иерархия классов, построенная на основе наследования) – иерархию объектов (иерархию типа целое/часть), построенную на основе агрегации.
Реализовать группу можно несколькими способами:
1. Класс “группа” содержит поля данных объектного типа. Таким образом, объект “группа” в качестве данных содержит либо непосредственно свои элементы, либо указатели на них

```
class TWindowDialog: public TGroup
 {
 protected:
 TInputLine input1;
 TEdit edit1;
 TButton button1;
 /*другие члены класса*/
 };
```

Такой способ реализации группы используется в C++Builder.

2. Группа содержит член-данное last типа TObject*, который указывает на начало связанного списка объектов, включенных в группу. В этом случае объекты должны иметь поле next типа TObject*, указывающее на следующий элемент в списке.Такой способ используется при реализации групп в Turbo Vision.

3. Создается связанный список структур типа TItem:

```
 struct TItem
 {TObject* item;
```

TItem* next;};
Поле item указывает на объект, включенный в группу. Группа содержит поле last типа TItem *, которое указывает на начало связанного списка структур типа TItem.
Если необходим доступ элементов группы к ее полям и методам, объект типа TObject должен иметь поле owner типа TGroup*, которое указывает на собственника этого элемента.

Методы группы.
Имеется два метода, которые необходимы для функционирования группы:
1) void Insert(TObject* p);
Вставляет элемент в группу.
2) void Show();
Позволяет просмотреть группу.
Кроме этого группа может содержать следующие методы:
1) int Empty();
Показывает, есть ли хотя бы один элемент в группе.
2) TObject* Delete(TObject* p);
Удаляет элемент из группы, но сохраняет его в памяти.
3) void DelDisp(TObject* p);
Удаляет элемент из группы и из памяти.

b. Иерархия объектов.

Иерархия классов есть иерархия по принципу наследования, т.е. типа "это есть разновидность того". Например, "рабочий есть разновидность персоны", "автомобиль" есть разновидность "транспортного средства". В отличие от этого и**ерархия объектов** – это иерархия по принципу вхождения, т.е. типа "это есть часть того". Например, "установка – часть завода", "двигатель" – часть "автомобиля". Таким образом, объекты нижнего уровня иерархии включаются в объекты более высокого уровня, которые являются для них группой.

c. Итератор.

Итераторы позволяют выполнять некоторые действия для каждого элемента определенного набора данных.
***For** all элементов набора { действия}*
Такой цикл мог бы быть выполнен для всего набора, например, чтобы напечатать все элементы набора, или мог бы искать некоторый элемент, который удовлетворяет определенному условию, и в этом случае такой цикл может закончиться, как только будет найден требуемый элемент.
Мы будем рассматривать итераторы как специальные методы класса-группы, позволяющие выполнять некоторые действия для всех объектов, включенных в группу. Примером итератора является метод *Show.*
Нам бы хотелось иметь такой итератор, который позволял бы выполнять над всеми элементами группы действия, заданные не одним из методов объекта, а произвольной функцией пользователя. Такой итератор можно реализовать, если эту функцию передавать ему через указатель на функцию.
Определим тип указателя на функцию следующим образом:

*typedef void(*PF)(TObject*,< дополнительные параметры>);*

Функция имеет обязательный параметр типа *TObject* или *TObject**, через который ей передается объект, для которого необходимо выполнить определенные действия.

Метод-итератор объявляется следующим образом:
void TGroup::ForEach(PF action,< дополнительные параметры >);

где
action – единственный обязательный параметр-указатель на функцию, которая вызывается для каждого элемента группы;
дополнительные параметры – передаваемые вызываемой функции параметры.
Затем определяется указатель на функцию и инициализируется передаваемой итератору функцией.

PF pf=myfunc;

Тогда итератор будет вызываться, например, для дополнительного параметра типа int, так:
gr.ForEach(pf,25);
Здесь gr – объект-группа.

d. Динамическая идентификация типов.

Динамическая идентификация типа характерна для языков, в которых поддерживается полиморфизм. В этих языках возможны ситуации, в которых тип объекта на этапе компиляции неизвестен.
В C++ полиморфизм поддерживается через иерархии классов, виртуальные функции и указатели базовых классов. При этом указатель базового класса может использоваться либо для указания на объект базового класса, либо для указания на объект любого класса, производного от этого базового.

Пусть группа содержит объекты различных классов и необходимо выполнить некоторые действия только для объектов определенного класса. Тогда в итераторе мы должны распознавать тип очередного объекта.
В стандарт языка C++ включены средства **RTTI** (Run-Time Type Idendification) – динамическая идентификация типов. Эти средства реализованы в последних системах Borland C++ (версий 4.0 и выше).
Информацию о типе объекта получают с помощью оператора typeid, определение которого содержит заголовочный файл <typeinfo.h>.
Имеется две формы оператора typeid:
typeid (объект)
typeid (имя_типа)
Оператор typeid возвращает ссылку на объект типа type_info.
В классе type_info перегруженные операции == и != обеспечивают сравнение типов.
Функция name () возвращает указатель на имя типа.
Имеется одно ограничение. Оператор typeid работает корректно только с объектами, у которых определены виртуальные функции. Большинство объектов имеют виртуальные функции, хотя бы потому, что обычно деструктор является виртуальным для устранения потенциальных проблем с производными классами. Когда оператор typeid применяют к неполиморфному классу (в классе нет виртуальной функции), получают указатель или ссылку базового типа.

Примеры.
1.
```
#include<iostream.h>
#include<typeinfo.h>
class Base{
virtual void f(){};
 //…
};
```

```
class Derived: public Base{
 //…
};
void main(){
 int i;
 Base ob,*p;
 Derived ob1;
 cout<<typeid(i).name(); //Выводится int
 p=&ob1;
 cout<<typeid(*p).name(); // Выводится Derived
}
```

2.

```
//начало см. выше
void WhatType(Base& ob){
 cout<< typeid(ob).name()<<endl;
}
void main(){
 Base ob;
 Derived ob1;
 WhatType(ob); //Выводится Base
 WhatType(ob1); //Выводится Derived
}
```

3.

```
//начало см. выше
void main(){
 Base *p;
 Derived ob;
 p=&ob;
 if(typeid(*p)==typeid(Derived)) cout<<“p указывает на объект типа Derived”;
 …
}
```

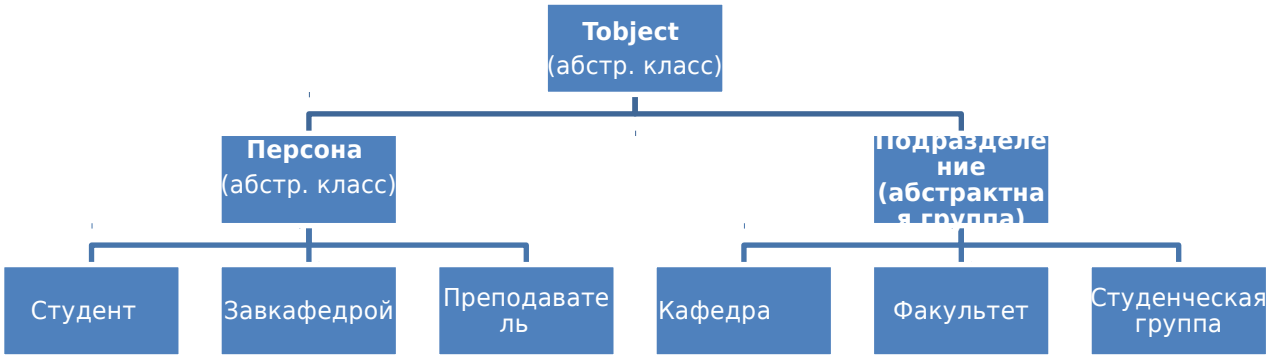
Если при обращении typeid(*p), p=NULL, то возбуждается исключительная ситуация bad_typeid

Порядок выполнения работы.

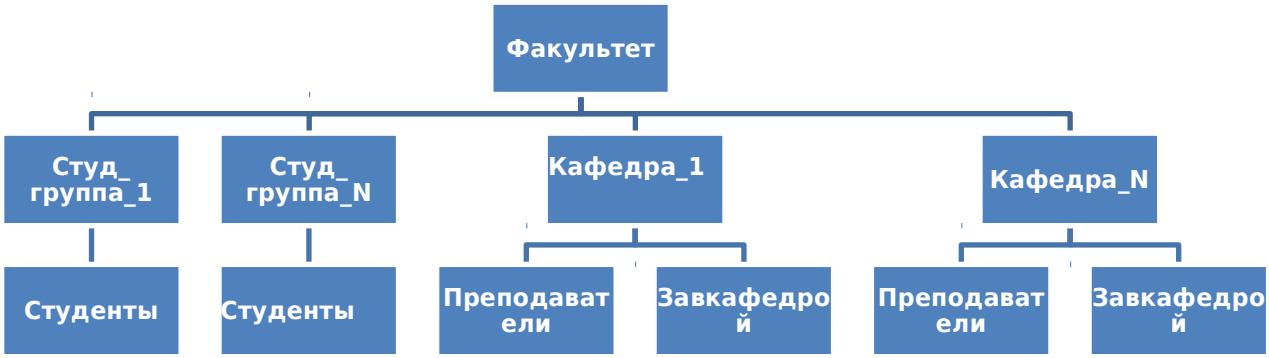
1. Дополнить иерархию классов лабораторной работы № 4 классами “группа”.
Например, для предметной области ФАКУЛЬТЕТ можно предложить классы “*факультет*”, “*студенческая группа*”, “*кафедра*”. Рекомендуется создать абстрактный класс – “*подразделение*”, который будет предком всех групп и абстрактный класс *TObject*, находящийся во главе всей иерархии.
2. Написать для класса-группы метод-итератор.
3. Написать процедуру или функцию, которая выполняется для всех объектов, входящих в группу (смотри Варианты запросов).
4. Написать демонстрационную программу, в которой создаются, показываются и разрушаются объекты-группы, а также демонстрируется использование итератора.

Методические указания.

1.Класс-группа должен соответствовать иерархии классов лабораторной работы № 5, т.е. объекты этих классов должны входить в группу. Например, для варианта 1 может быть предложена следующая иерархия классов:



При этом иерархия объектов будет иметь следующий вид:



2. Для включения объектов в группу следует использовать третий способ (через связанный список структур типа TItem).

3. Пример определения добавленных абстрактных классов:

```
class TObject
{
public:
virtual void Show()=0;};

class TDepartment:public TObject // абстрактный класс-группа
{
protected:
char name[20]; // наименование
TPerson* head; // руководитель
TItem* last; // указатель на начало связанного списка структур TItem
public:
TDepartment(char*,TPerson*);
TDepardment(TDepartment&);
~ TDepartment();
char* GetName();
TPerson* GetHead();
void SetName(char* NAME);
void SetHead(TPerson* p);
void Insert(TObject* p);
virtual void Show()=0;
};
```

4. Иерархия объектов создается следующим образом (на примере ФАКУЛЬТЕТА):
а) создается пустой ФАКУЛЬТЕТ,
б) создается пустая КАФЕДРА,
в) создаются ПРЕПОДАВАТЕЛИ и включаются в КАФЕДРУ,
г) КАФЕДРА включается в ФАКУЛЬТЕТ,
д) тоже повторяется для другой кафедры,
е) создается пустая СТУДЕНЧЕСКАЯ ГРУППА,
ж) создаются СТУДЕНТЫ и включаются в СТУДЕНЧЕСКУЮ ГРУППУ,
з) СТУДЕНЧЕСКАЯ ГРУППА включается в ФАКУЛЬТЕТ,
и) тоже повторяется для другой студенческой группы.
5. Удаляется ФАКУЛЬТЕТ (при вызове деструктора) в обратном порядке.
6. Метод-итератор определяется в неабстрактных классах-группах на основе выбранных запросов.
Например, для класса *TStudentGroup* может быть предложен итератор *void TStudentGroup::ForEach(PF action, float parametr);*
где *action* – указатель на функцию, которая должна быть выполнена для всех объектов, включенных в группу (в данном случае для всех ОБУЧАЮЩИХСЯ), *parametr*- передаваемая процедуре дополнительная информация.
В качестве передаваемой методу функции может быть предложена, например, такая: вывести список обучающихся, имеющих рейтинг не ниже заданного

```
void MyProc(TObject* p,float rate)
{
if (((TStudent*)p) ->GetGrade()>=rate) cout<<(((TStudent*)p) ->GetName());
}
```

7. Студент определяет передаваемую итератору функции на основе запросов, которые должны быть выполнены вызовом итератора. Варианты запросов приведены в приложении.

Варианты запросов.

1. Имена всех лиц мужского (женского) пола.
2. Имена обучающихся указанного курса.
3. Имена и должность преподавателей указанной кафедры.
4. Имена служащих со стажем не менее заданного.
5. Имена служащих заданной профессии.
6. Имена рабочих заданного цеха.
7. Имена рабочих заданной профессии.
8. Имена обучающихся, сдавших все (заданный) экзамены на отлично (хорошо и отлично).
9. Имена обучающихся, не сдавших все (хотя бы один) экзамен.
10. Имена всех монархов на заданном континенте.
11. Наименование всех деталей (узлов), входящих в заданный узел (механизм).
12. Наименование всех книг в библиотеке (магазине), вышедших не ранее указанного года.
13. Названия всех городов заданной области.
14. Наименование всех товаров в заданном отделе магазина.
15. Количество мужчин (женщин).
16. Количество обучающихся на указанном курсе.
17. Количество служащих со стажем не менее заданного.
18. Количество рабочих заданной профессии.
19. Количество инженеров в заданном подразделении.
20. Количество товара заданного наименования.
21. Количество обучающихся, сдавших все экзамены на отлично.
22. Количество обучающихся, не сдавших хотя бы один экзамен.
23. Количество деталей (узлов), входящих в заданный узел (механизм).

24. Количество указанного транспортного средства в автопарке (на автостоянке).
25. Количество пассажиров во всех вагонах экспресса.
26. Суммарная стоимость товара заданного наименования.
27. Средний балл за сессию заданного студента.
28. Средний балл по предмету для всех обучающихся.
29. Суммарное количество учебников в библиотеке (магазине).
30. Суммарное количество жителей всех городов в области.
31. Суммарная стоимость продукции заданного наименования по всем накладным.
32. Средняя мощность всех (заданного типа) транспортных средств в организации.
33. Средняя мощность всех дизелей, обслуживаемых заданной фирмой.
34. Средний вес животных заданного вида в зоопарке.
35. Среднее водоизмещение всех парусников на верфи (в порту).