МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное учреждение
высшего профессионального образования
"Казанский (Приволжский) федеральный университет"
Институт математики и механики им. Н.И. Лобачевского

подписано электронно-цифровой подписью

Программа дисциплины

Технологии программирования и работа на электронных вычислительных машинах Б2.Б.4

Направление подготовки: 010800.62 - Механика и математическое моделирование
Профиль подготовки: Общий профиль
Квалификация выпускника: бакалавр
Форма обучения: очное
Язык обучения: русский

Автор(ы):

Липачев Е.К.

Рецензент(ы):

Маклецов С.В.

СОГЛАСОВАНО:

Заведующий(ая) кафедрой: Авхадиев Ф. Г.
Протокол заседания кафедры No ___ от "____" _____ 201__г

Учебно-методическая комиссия Института математики и механики им. Н.И. Лобачевского :
Протокол заседания УМК No ____ от "____" _____ 201__г

Регистрационный No 8172315

Казань
2014

Содержание

Программу дисциплины разработал(а)(и) доцент, к.н. (доцент) Липачев Е.К. Кафедра теории функций и приближений отделение математики , Evgeny.Lipachev@kpfu.ru

1. Цели освоения дисциплины

Целями освоения дисциплины Технология программирования и работа на ЭВМ являются понимание основных идей, лежащих в основе компьютерных наук, их практическое применение и возможности; владение теоретическими знаниями основных методов алгоритмизации прикладных задач математики, механики, физики и других наук; ориентирование в потоке информации о компьютерных науках; приобретение навыков применения методов компьютерных наук для различного класса задач, умения довести их до числа

2. Место дисциплины в структуре основной образовательной программы высшего профессионального образования

Данная учебная дисциплина включена в раздел " Б2.Б.4 Общепрофессиональный" основной образовательной программы 010800.62 Механика и математическое моделирование и относится к базовой (общепрофессиональной) части. Осваивается на 1, 2 курсах, 1, 2, 3, 4 семестры.

Дисциплина "Технология программирования и работа на ЭВМ" относится к блоку Б2. Для освоения требуется знание школьного курса математики и информатики.

3 Компетенции обучающегося, формируемые в результате освоения дисциплины (модуля) ОК 6, 10, 12, 13, ПК 1-3, 5, 7-11.

В результате освоения дисциплины обучающийся должен:

- Знать основные идеи, лежащие в основе компьютерных наук, их практическое применение и возможности; основные методы алгоритмизации прикладных задач математики, механики, физики и других наук;

- Уметь: находить, анализировать и контекстно-обрабатывать навучно-техническую информацию с помощью компьютера; активно использовать компьютер в профессиональной и социально-бытовой сфере; создавать базы данных и использовать ресурсы Интернет.

- Владеть навыками применения методов компьютерных наук для различного класса задач, умением довести их до числа; базовыми знаниями в областях информатики и информационных технологий, навыками использования программных средств и работы в компьютерных сетях.

3. Компетенции обучающегося, формируемые в результате освоения дисциплины /модуля

В результате освоения дисциплины формируются следующие компетенции:

Шифр компетенции	Расшифровка приобретаемой компетенции
ОК 6 (общекультурные компетенции)	ОК 6 способность работать самостоятельно
ОК-10 (общекультурные компетенции)	ОК-10 умением находить, анализировать и контекстно обрабатывать научно-техническую информацию (ОК-10);
ОК-12 (общекультурные компетенции)	ОК-12 навыками работы с компьютером

Шифр компетенции	Расшифровка приобретаемой компетенции
ОК-13 (общекультурные компетенции)	ОК-13 базовыми знаниями в областях информатики и современных информационных технологий, навыки использования программных средств и навыки работы в компьютерных сетях, умение создавать базы данных и использовать ресурсы Интернет
ПК-1 (профессиональные компетенции)	ПК-1 владение методами математического моделирования при анализе проблем на основе знаний фундаментальных дисциплин
ПК2 (профессиональные компетенции)	ПК-2 умением понять поставленную задачу (ПК-2);
ПК-3 (профессиональные компетенции)	ПК-3 умением формулировать результат (ПК-3);
ПК-5 (профессиональные компетенции)	ПК-5 умение представить публично собственные результаты

В результате освоения дисциплины студент:

1. должен знать:

Знать основные идеи, лежащие в основе компьютерных наук, их практическое применение и возможности; основные методы алгоритмизации прикладных задач математики, механики, физики и других наук;

2. должен уметь:

находить, анализировать и контекстно-обрабатывать навучно-техническую информацию с помощью компьютера; активно использовать компьютер в профессиональной и социально-бытовой сфере; создавать базы данных и использовать ресурсы Интернет.

3. должен владеть:

Владеть навыками применения методов компьютерных наук для различного класса задач, умением довести их до числа; базовыми знаниями в областях информатики и информационных технологий, навыками использования программных средств и работы в компьютерных сетях.

4. должен демонстрировать способность и готовность:

В результате освоения дисциплины обучающийся должен:

- Знать основные идеи, лежащие в основе компьютерных наук, их практическое применение и возможности; основные методы алгоритмизации прикладных задач математики, механики, физики и других наук;
- Уметь: находить, анализировать и контекстно-обрабатывать навучно-техническую информацию с помощью компьютера; активно использовать компьютер в профессиональной и социально-бытовой сфере; создавать базы данных и использовать ресурсы Интернет.
- Владеть навыками применения методов компьютерных наук для различного класса задач, умением довести их до числа; базовыми знаниями в областях информатики и информационных технологий, навыками использования программных средств и работы в компьютерных сетях.

4. Структура и содержание дисциплины/ модуля

Общая трудоемкость дисциплины составляет 14 зачетных(ые) единиц(ы) 504 часа(ов).

Форма промежуточного контроля дисциплины экзамен в 1 семестре; зачет во 2 семестре; зачет в 3 семестре; экзамен в 4 семестре.

Суммарно по дисциплине можно получить 100 баллов, из них текущая работа оценивается в 50 баллов, итоговая форма контроля - в 50 баллов. Минимальное количество для допуска к зачету 28 баллов.

86 баллов и более - "отлично" (отл.);

71-85 баллов - "хорошо" (хор.);

55-70 баллов - "удовлетворительно" (удов.);

54 балла и менее - "неудовлетворительно" (неуд.).

4.1 Структура и содержание аудиторной работы по дисциплине/ модулю

Тематический план дисциплины/модуля

N	Раздел Дисциплины/ Модуля	Семестр	Неделя семестра	Виды и часы аудиторной работы, их трудоемкость (в часах)			Текущие формы контроля
				Лекции	Практические занятия	Лабораторные работы	
1.	Тема 1. История развития вычислительной техники, ее современное состояние	1		2	0	0	дискуссия
2.	Тема 2. ЗАКОНЫ РАЗВИТИЯ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ	1		4	0	0	дискуссия
3.	Тема 3. КОМПЬЮТЕРНЫЕ СЕТИ. Адресация.	1		4	0	4	контрольная работа
4.	Тема 4. Интернет-технологии. Общие понятия.	1		4	0	4	устный опрос

N	Раздел Дисциплины/ Модуля	Семестр	Неделя семестра	Виды и часы аудиторной работы, их трудоемкость (в часах)			Текущие формы контроля
				Лекции	Практические занятия	Лабораторные работы	
5.	Тема 5. Современный графический интерфейс пользователя (ГИП). Понятие графического компонента. Окно, типы окон, элементы окна, взаимное расположение окон, их перемещение, изменение размеров, пиктограммы. Рабочий стол. Запуск, переключение и завершение программ. Настройка рабочего стола. Работа с папками. Настройка параметров ГИП, экрана, мыши и клавиатуры. Шрифты, их установка. Работа с документами при помощи мыши и клавиатуры, обмен данными через буфер	1		2	0	2	устный опрос
6.	Тема 6. Текстовые процессоры, основные приемы работы с ними. Шрифты, гарнитуры, их элементы и типы. Набор формул. Работа с таблицами и формами. Вставка и создание графики. Понятие стиля и шаблона. Драйверы принтеров, их установка и настройка, управление процессом печати.	1		2	0	2	домашнее задание

N	Раздел Дисциплины/ Модуля	Семестр	Неделя семестра	Виды и часы аудиторной работы, их трудоемкость (в часах)			Текущие формы контроля
				Лекции	Практические занятия	Лабораторные работы	
7.	Тема 7. Электронные таблицы, основные понятия. Рабочая книга, рабочий лист, ячейка, работа с ними. Создание формул, массивов и функций. Представление данных графиками и диаграммами. Анализ данных, итоги и сводные таблицы.	1		2	0	4	домашнее задание
8.	Тема 8. Математические пакеты, их основные возможности. Вычисление выражений и функций, символьные преобразования, построение графиков. Встроенный язык программирования. Расширение ядра с помощью встраиваемых модулей и пакетов.	1		2	0	6	домашнее задание
9.	Тема 9. Системы счисления: позиционные и непозиционные. Примеры систем счисления.	1		2	0	2	контрольная работа
10.	Тема 10. Алгоритм. Основные особенности алгоритма.	1		2	0	2	устный опрос
11.	Тема 11. Запись алгоритмов. Блок-схемы.	1		4	0	2	домашнее задание
12.	Тема 12. Рекуррентные соотношения. Вычисление рекуррентных соотношений.	1		2	0	2	контрольная работа
13.	Тема 13. Алгоритм вычисление сумм и произведений.	1		2	0	2	домашнее задание

N	Раздел Дисциплины/ Модуля	Семестр	Неделя семестра	Виды и часы аудиторной работы, их трудоемкость (в часах)			Текущие формы контроля
				Лекции	Практические занятия	Лабораторные работы	
14.	Тема 14. Алгоритмы вычисления минимальных и максимальных значений. Алгоритм вычисления количества элементов последовательности с заданным свойством.	1		2	0	4	домашнее задание
15.	Тема 15. Синтаксис языка С. Операции арифметические, сравнения и логические, приоритет выполнения операций. Операции инкремента (++) и декремента (--). Оператор запятая. Оператор "?". Логические операции и оператор if.Оператор switch.Операторы цикла for(), while.	2		4	0	2	устный опрос
16.	Тема 16. Массивы. Многомерные массивы	2		2	0	2	домашнее задание
17.	Тема 17. Динамические массивы. Создание массивов с помощью malloc(). Создание массивов с помощью new.	2		2	0	2	домашнее задание
18.	Тема 18. Арифметика указателей. Указатели и массивы.	2		4	0	4	домашнее задание

N	Раздел Дисциплины/ Модуля	Семестр	Неделя семестра	Виды и часы аудиторной работы, их трудоемкость (в часах)			Текущие формы контроля
				Лекции	Практические занятия	Лабораторные работы	
19.	Тема 19. Функции - правила создания. Объявление и определение функции. Формальные и фактические параметры. Передача параметров функции по значению и по ссылке. Массивы в качестве параметров функции. Аргументы по умолчанию. Рекурсия. Правила перегрузки функций. Указатель на функцию. Имя функции как параметр функции.	2		4	0	4	домашнее задание
20.	Тема 20. Файлы - общие правила. Запись в текстовый файл. Чтение из текстового файла.	2		4	0	4	домашнее задание
21.	Тема 21. Структуры.	2		2	0	2	устный опрос
22.	Тема 22. Алгоритмы приближенного решения уравнений.	2		2	0	2	домашнее задание
23.	Тема 23. Алгоритмы работы с матрицами.	2		4	0	4	домашнее задание
24.	Тема 24. Алгоритмы решения систем линейных уравнений. Реализация метода исключения Гаусса. Вычисление определителей. Вычисление обратной матрицы.	2		4	0	6	домашнее задание
25.	Тема 25. Повышение точности вычислений.	2		2	0	2	дискуссия
26.	Тема 26. Основные понятия объектно-ориентированного программирования (ООП).	3		2	0	2	

N	Раздел Дисциплины/ Модуля	Семестр	Неделя семестра	Виды и часы аудиторной работы, их трудоемкость (в часах)			Текущие формы контроля
				Лекции	Практические занятия	Лабораторные работы	
27.	Тема 27. Классы. Атрибуты и операции. Реализация операций (функций-членов) класса вне класса. Инкапсуляция. Модификаторы доступа private и public.	3		4	0	4	дискуссия
28.	Тема 28. Конструкторы и деструкторы. Конструктор по умолчанию. Конструктор копирования.	3		4	0	4	домашнее задание
29.	Тема 29. Объекты. Массив объектов. Инициализация массива объектов. динамическое выделение памяти объектам. Указатели на объекты. Указатель this. Передача объектов функции по ссылке и по значению.Дружественные функции. Возвращение объекта из функции.	3		4	0	4	домашнее задание
30.	Тема 30. . Наследование. Правила наследования с модификаторами private и public. Модификатор доступа protected. Конструкторы и наследование. Вызов методов Множественное наследование.базового класса в производном.	3		6	0	6	домашнее задание
31.	Тема 31. Виртуальные методы. Статическое и динамическое связывание.	3		4	0	4	дискуссия
32.	Тема 32. Абстрактные классы.	3		2	0	2	дискуссия

N	Раздел Дисциплины/ Модуля	Семестр	Неделя семестра	Виды и часы аудиторной работы, их трудоемкость (в часах)			Текущие формы контроля
				Лекции	Практические занятия	Лабораторные работы	
33.	Тема 33. Константные переменные. Указатель на константу. Константные параметры функций.	3		2	0	2	дискуссия
34.	Тема 34. Операции с файлами в C++ (классы ofstream и ifstream). Чтение из файла, запись в файл.	3		2	0	2	
35.	Тема 35. Перегрузка операций.	3		2	0	2	домашнее задание
36.	Тема 36. Раздельная компиляция, "стражи включения".	3		2	0	2	
37.	Тема 37. Пространства имен.	3		2	0	2	дискуссия
38.	Тема 38. Структуры данных. Внутренняя и внешняя сортировки	4		2	0	0	
39.	Тема 39. Сортировка числового массива методом пузырька. Сортировка массива объектов методом пузырька. Сортировка числового массива методом просеивания. Сортировка числового массива методом Шелла. Сортировка числового массива методом быстрой сортировки.	4		6	6	0	
40.	Тема 40. Метод естественного слияния.	4		2	2	0	
41.	Тема 41. Алгоритмы внешней сортировки.	4		6	8	0	дискуссия
42.	Тема 42. Алгоритмы поиска	4		4	4	0	
43.	Тема 43. Деревья цифрового поиска	4		4	4	0	

N	Раздел Дисциплины/ Модуля	Семестр	Неделя семестра	Виды и часы аудиторной работы, их трудоемкость (в часах)			Текущие формы контроля
				Лекции	Практические занятия	Лабораторные работы	
44.	Тема 44. Создание однонаправленного списка. Создание двунаправленного списка. Добавление нового узла в начало однонаправленного списка. Добавление нового узла в начало двунаправленного списка. Добавление нового узла в конец однонаправленного списка. Добавление нового узла в конец двунаправленного списка. Добавление нового узла в определенное место однонаправленного списка. Добавление нового узла в определенное место двунаправленного списка. Удаление первого узла однонаправленного списка. Удаление первого узла двунаправленного списка. Удаление последнего узла однонаправленного списка. Удаление последнего узла двунаправленного списка. Удаление заданного узла однонаправленного списка. Удаление заданного узла двунаправленного списка.	4		6	6	0	
45.	Тема 45. Программирование в среде Visual Studio. Создание приложений с графическим интерфейсом	4		4	4	0	
.	Тема . Итоговая форма контроля	1		0	0	0	экзамен

N	Раздел Дисциплины/ Модуля	Семестр	Неделя семестра	Виды и часы аудиторной работы, их трудоемкость (в часах)			Текущие формы контроля
				Лекции	Практические занятия	Лабораторные работы	
.	Тема . Итоговая форма контроля	2		0	0	0	зачет
.	Тема . Итоговая форма контроля	3		0	0	0	зачет
.	Тема . Итоговая форма контроля	4		0	0	0	экзамен
	Итого			140	34	106	

4.2 Содержание дисциплины

Тема 1. История развития вычислительной техники, ее современное состояние

лекционное занятие (2 часа(ов)):

Этапы развития компьютерных технологий. ЭВМ в СССР и мире.

Тема 2. ЗАКОНЫ РАЗВИТИЯ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

лекционное занятие (4 часа(ов)):

Закон Мура Закон Рока Закон Крайдера Закон Белла Закон Меткалфа Закон Рида Закон Ципфа

Тема 3. КОМПЬЮТЕРНЫЕ СЕТИ. Адресация.

лекционное занятие (4 часа(ов)):

Топология компьютерных сетей.

лабораторная работа (4 часа(ов)):

TCP/IP. Классы сетей.

Тема 4. Интернет-технологии. Общие понятия.

лекционное занятие (4 часа(ов)):

WWW. URI. URN

лабораторная работа (4 часа(ов)):

HTML.

Тема 5. Современный графический интерфейс пользователя (ГИП). Понятие графического компонента. Окно, типы окон, элементы окна, взаимное расположение окон, их перемещение, изменение размеров, пиктограммы. Рабочий стол. Запуск, переключение и завершение программ. Настройка рабочего стола. Работа с папками. Настройка параметров ГИП, экрана, мыши и клавиатуры. Шрифты, их установка. Работа с документами при помощи мыши и клавиатуры, обмен данными через буфер

лекционное занятие (2 часа(ов)):

Операционная система Windows. Окно, типы окон, элементы окна, взаимное расположение окон, их перемещение, изменение размеров, пиктограммы. Рабочий стол. Запуск, переключение и завершение программ.

лабораторная работа (2 часа(ов)):

Выполнение практических операций с окнами

Тема 6. Текстовые процессоры, основные приемы работы с ними. Шрифты, гарнитуры, их элементы и типы. Набор формул. Работа с таблицами и формами. Вставка и создание графики. Понятие стиля и шаблона. Драйверы принтеров, их установка и настройка, управление процессом печати.

лекционное занятие (2 часа(ов)):

MS Word, LibreOffice

лабораторная работа (2 часа(ов)):

Выполнение практических операций

Тема 7. Электронные таблицы, основные понятия. Рабочая книга, рабочий лист, ячейка, работа с ними. Создание формул, массивов и функций. Представление данных графиками и диаграммами. Анализ данных, итоги и сводные таблицы.

лекционное занятие (2 часа(ов)):

MS Excel

лабораторная работа (4 часа(ов)):

Выполнение практических операций

Тема 8. Математические пакеты, их основные возможности. Вычисление выражений и функций, символьные преобразования, построение графиков. Встроенный язык программирования. Расширение ядра с помощью встраиваемых модулей и пакетов.

лекционное занятие (2 часа(ов)):

Mathematica, MatLab

лабораторная работа (6 часа(ов)):

Выполнение практических операций

Тема 9. Системы счисления: позиционные и непозиционные. Примеры систем счисления.

лекционное занятие (2 часа(ов)):

Позициорнные и непозиционные системы счисления. Перевод из одной системы счисления в другую. Двоичная система счисления. Шестнадцатеричная система счисления. Единицы измерения памяти (байт, кибибайт и т.д.).

лабораторная работа (2 часа(ов)):

Перевод из одной системы счисления в другую.

Тема 10. Алгоритм. Основные особенности алгоритма.

лекционное занятие (2 часа(ов)):

Основные этапы полного построения алгоритма. 1. Постановка задачи. 2. Построение модели. 3. Разработка алгоритма. 4. Проверка правильности алгоритма. 5. Реализация алгоритма. 6. Анализ алгоритма и его сложности. 7. Проверка программы. 8. Составление документации.

лабораторная работа (2 часа(ов)):

Примеры алгоритмов.

Тема 11. Запись алгоритмов. Блок-схемы.

лекционное занятие (4 часа(ов)):

Один из способов записи алгоритмов основан на использовании блок ? схем. Любая блок схема может быть построена из четырёх элементарных блок ? схем. Первая элементарная блок?схема называется композицией, вторая ? выбором или развилкой, третья и четвертая блок?схемы представляют цикл с предусловием и постусловием, соответственно.

лабораторная работа (2 часа(ов)):

Как создавать блок-схемы. Программные инструменты.

Тема 12. Рекуррентные соотношения. Вычисление рекуррентных соотношений.

лекционное занятие (2 часа(ов)):

Рекуррентные соотношения выражают значения функции при помощи других значений, вычисленных для меньших аргументов. Пример рекуррентной последовательности ? последовательность чисел Фибоначчи (см., например, Кнут Д. Искусство программирования для ЭВМ. Т. 1. Основные алгоритмы. ? М.: Мир, 1976., стр. 112; Грэхем Р., Кнут Д., Паташник О. Конкретная математика. Основания информатики. ? М.: Мир, 1998., стр. 322). Вместо термина рекуррентное соотношение, говорят также о возвратном соотношении и рекурсивной зависимости. Решение рекуррентного соотношения ? это нахождение выражения для n-го члена последовательности в замкнутой форме, позволяющей найти значение каждого члена последовательности без вычисления предыдущих членов. Подробности см. в книге Грэхем Р., Кнут Д., Паташник О. Конкретная математика. Основания информатики. ? М.: Мир, 1998. ? 703 с. Сейчас остановимся только на алгоритмической стороне.

лабораторная работа (2 часа(ов)):

Пример. Вводится целое положительное число n. Вычислить n-й член последовательности Фибоначчи. Пример. Приближённое вычисление квадратного корня. Вводится положительное действительное число a. Значение квадратного корня из числа a с заданной точностью (например, с 5-ю знаками) можно получить с помощью следующего рекуррентного соотношения x0 = a, xk = 0.5 (xk-1 + a/xk-1), k=1,2,3, ?. Процесс вычислений заканчивается, если |xk ? xk-1|<10-5. Пример. Факториал. Пример. Схема Горнера вычисления значения многочлена n-й степени

Тема 13. Алгоритм вычисление сумм и произведений.

лекционное занятие (2 часа(ов)):

Сумма как рекуррентная последовательность. Вычисление сумм. От блок-схем к программам. Произведение как рекуррентная последовательность. Бесконечные суммы См. Кнут Д. Искусство программирования для ЭВМ. Т. 1. Основные алгоритмы. ? М.: Мир, 1976., стр. 58. Вычислить бесконечную сумму с заданной точностью Считать, что требуемая точность достигнута, если вычислена сумма нескольких первых слагаемых и очередное слагаемое оказалось по модулю меньше, чем , ? это и последующие слагаемые можно уже не учитывать.

лабораторная работа (2 часа(ов)):

Составление блок-схем вычисления сумм и произведений.

Тема 14. Алгоритмы вычисления минимальных и максимальных значений. Алгоритм вычисления количества элементов последовательности с заданным свойством.

лекционное занятие (2 часа(ов)):

Вычисление максимальных и минимальных значений в виде рекуррентных соотношений.

лабораторная работа (4 часа(ов)):

Блок-схемы и программы вычисления минимальных значений числовых последовательностей.

Тема 15. Синтаксис языка С. Операции арифметические, сравнения и логические, приоритет выполнения операций. Операции инкремента (++) и декремента (--). Оператор запятая. Оператор "?". Логические операции и оператор if.Оператор switch.Операторы цикла for(), while.

лекционное занятие (4 часа(ов)):

Программа С строится из отдельных блоков, называемых функциями. Каждая функция состоит из операторов. Оператор является завершённой инструкцией для компьютера. Комментарии. Препроцессор. Представление данных. Операции. Операции сравнения и логические операции. Операции присваивания. Операции инкремента и декремента. Оператор ?запятая?. Одно выражение может состоять из набора подвыражений, разделенных запятыми; такие подвыражения вычисляются слева направо. Конечным результатом будет результат самого правого из них. Приоритеты операций задают последовательность вычислений в сложном выражении. В С++ умножение и деление имеют более высокий приоритет, чем сложение, поэтому они будут вычислены раньше. Их собственные приоритеты равны, поэтому умножение и деление будут вычисляться слева направо. Инструкции. Инструкция if обеспечивает выполнение или пропуск инструкции или блока в зависимости от условия. Вместо оператора if-else часто используется оператор ?:. выражение1 ? выражение2 : выражение3 Если выражение1 истинно, то значение всего оператора равно значению выражение2, в противном случае значением всего выражения будет выражение3. Инструкция switch состоит из следующих частей: ? ключевого слова switch, за которым в круглых скобках идет выражение, ? набора меток case, состоящих из ключевого слова case и константного выражения, с которым сравнивается условие. ? последовательности инструкций, соотносимых с метками case. ? необязательной метки default, которая является аналогом части else инструкции if-else. Инструкции, соответствующие этой метке, выполняются, если условие не отвечает ни одной из меток case. Инструкция цикла for. Инструкция while while (условие) оператор Пока условие является истинным, инструкция выполняется в такой последовательности: 1. Вычислить условие. 2. Выполнить оператор, если условие истинно. 3. Если самое первое вычисление условия дает false, оператор не выполняется. Инструкция do while Использование do while, гарантирует выполнение тела цикла хотя бы один раз. Синтаксис цикла do while do оператор while (условие); инструкция выполняется до первой проверки условия. Если вычисление условия дает false, цикл останавливается. Инструкция break Инструкция break останавливает циклы for, while, do while и блока switch. Выполнение программы продолжается с инструкции, следующей за закрывающей фигурной скобкой цикла или блока. Инструкция continue Инструкция continue завершает текущую итерацию цикла и передает управление на вычисление условия, после чего цикл может продолжиться. В отличие от инструкции break, завершающей выполнение всего цикла, инструкция continue завершает выполнение только текущей итерации. Инструкция goto обеспечивает безусловный переход к другой инструкции внутри той же функции, поэтому современная практика программирования выступает против ее применения.

лабораторная работа (2 часа(ов)):

Упражнения на синтаксические конструкции языка программирования.

Тема 16. Массивы. Многомерные массивы

лекционное занятие (2 часа(ов)):

Массив ? это набор данных одного типа. Для создания массива используется оператор объявления. Объявление массива состоит из 3 частей: тип элементов массива, имя массива, количество элементов в массиве: Type arrayName[arraySize]; Выражение arraySize, которое задает количество элементов, должно быть константным (10, 20 и т.д.), либо постоянным выражением, значение которого известно во время компиляции, например, 10. Если необходимо создать массив, число элементов которого будет установлено только во время выполнения программы, то должны использоваться либо функция malloc() или оператор new, ? это будет показано в разделе ?Динамические массивы?. Нельзя присваивать одну переменную-массив другой, недопустимо их сравнивать, складывать и т. д. Все операции можно выполнять только с отдельными элементами массива. В С и С++ поддерживаются многомерные массивы. При определении многомерных массивов задается тип элементов массива, имя массива, а затем, в отдельных квадратных скобках, число элементов в первой размерности, в отдельных квадратных скобках ? число элементов во второй размерности и т.д. Многомерные массивы можно инициализировать, используя синтаксис, аналогичный одномерному случаю.

лабораторная работа (2 часа(ов)):

Упражнения на использование массивов.

Тема 17. Динамические массивы. Создание массивов с помощью malloc(). Создание массивов с помощью new.

лекционное занятие (2 часа(ов)):

При создании указателя память выделяется только для хранения адреса. Для выделения памяти под данные используется оператор new (или функция malloc()). Для динамически размещаемого одномерного массива используется следующая форма оператора p=new type [size] для удаления динамически размещаемого одномерного массива используется delete [] p; Выделение памяти для массива в процессе компиляции называется статическим связыванием. Память под массив выделяется на этапе компиляции. С помощью оператора new можем создавать массив во время выполнения программы, размер массива также определяется на этапе выполнения программы. Такой массив называется динамическим, а процесс его создания ? динамическим связыванием.

лабораторная работа (2 часа(ов)):

Упражнения на динамические массивы.

Тема 18. Арифметика указателей. Указатели и массивы.

лекционное занятие (4 часа(ов)):

Указатель ? это переменная, содержащая адрес другой переменной. Применяя операцию * (операция разыменования), получаем значение, записанное по данному адресу. С помощью операции &, применённой к переменной, можно узнать адрес, по которому эта переменная хранится в памяти. При объявлении указателя также используется ?*?, кроме того, необходимо определить на какой тип данных ссылается указатель. При создании указателя память выделяется только для хранения адреса. Для выделения памяти под данные используется оператор new (или функция malloc()). Арифметика указателей разрешает вычитать целые числа из указателей, в результате получаем адрес, уменьшенный на количество байт, необходимых для размещения этого числа элементов. К указателям применимы операции ++. Кроме того, можем вычитать один указатель из другого, в случае, если они являются указателями на один и тот же тип данных. Эта разность равна количеству элементов данного типа, размещённых между двумя адресами. Указатели и массивы тесно связаны. Имя массива является адресом первого элемента этого массива. Но между обычным указателем и массивом есть одно существенное различие ? значение указателя можно изменять, а имя массива является константой.

лабораторная работа (4 часа(ов)):

Упражнения на использование указателей.

Тема 19. Функции - правила создания. Объявление и определение функции. Формальные и фактические параметры. Передача параметров функции по значению и по ссылке. Массивы в качестве параметров функции. Аргументы по умолчанию. Рекурсия. Правила перегрузки функций. Указатель на функцию. Имя функции как параметр функции.

лекционное занятие (4 часа(ов)):

Разделение программы на отдельные логические блоки ? функции, позволяет эффективнее управлять кодом. Язык программирования предлагает средства коммуникации с функциями ? методы передачи параметров и возврата значений из функций. Использование функции предполагает: ? объявление функции (её прототип), включающее имя функции, возвращаемый ею тип, типы её параметров; ? определение функции, включающее заголовок функции и последовательность операторов, реализующих функцию; ? вызов функции, указанием ее имени и заменой параметров значениями. До обработки вызова функции компилятору должно быть известно её объявление. Прототип функции является оператором, поэтому он должен заканчиваться точкой с запятой. В прототипе функции не требуется указывать имена параметров, ? достаточно списка типов. В определении функции реализуется алгоритм функции. Определение начинается с заголовка функции. В отличие от прототипа функции, заголовок не должен заканчиваться точкой с запятой. Кроме того, в заголовке обязательно указываются имена параметров. Функция должна содержать хотя бы один оператор return ? исключением являются функции, у которых тип возвращаемых значений void. Выполнение оператора return завершает работу функции и возвращает управление в вызывающую программу. return выражение; Для функции с типом возвращения void оператор return необязателен, но его можно использовать и для таких функций, для завершения работы функции и передачи управления в вызывающую программу. Синтаксис оператора, в этом случае, следующий return; Переменная, которая используется для приёма передаваемого значения, называется формальным параметром. Величина, передаваемая функции, называется фактическим параметром. Переменные, в том числе и параметры функции, действуют только в пределах данной функции. Во время вызова функции компилятор выделяет память для этих переменных, а по завершению выполнения функции, память, задействованная этими переменными, освобождается. В языке C++ введён новый составной тип данных ? ссылочная переменная. Ссылка представляет собой имя, которое является псевдонимом для ранее объявленной переменной. Для объявления ссылочной переменной используется символ &. Основное назначение ссылок ? использование в качестве формальных параметров функций. Используя ссылку в качестве аргумента, функция работает с исходными данными, а не с их копиями. В языке C этого средства не было, появилось в C++. При объявлении функции можно задать значения по умолчанию для одного или нескольких параметров в списке. Рекурсия. Функция может вызывать саму себя. Перегрузка функций (полиморфизм функций) реализована в C++, в языке C такой возможности не было. Полиморфизм функций позволяет использовать под одним именем несколько функций. Главную роль в перегрузке функций играет список аргументов, который называют также сигнатурой функции. Если две функции имеют одно и то же количество аргументов, эти аргументы имеют одинаковые типы и порядок следования, то функции, по определению, имеют одинаковую сигнатуру. В языке C++ можно определить две различные функции с одинаковым именем при условии, если эти функции обладают различными сигнатурами. Указатели на функцию. Функции, как и переменные, имеют адреса. Адресом функции является адрес памяти, с которого начинается машинный код функции.

лабораторная работа (4 часа(ов)):

Упражнения на использование функций. Использование массива как параметра функции. Пример аргументов по умолчанию. Аргумент по умолчанию позволяет пропускать аргументы при вызове функции. Пропущеный аргумент получит значение по умолчанию. Пример. Перегрузка функции min().

Тема 20. Файлы - общие правила. Запись в текстовый файл. Чтение из текстового файла.

лекционное занятие (4 часа(ов)):

Завершение работы программы приводит к освобождению памяти ? значения всех переменных теряются. Для сохранения информации используются файлы. Файл представляет собой поток байтов. Каждый файл заканчивается маркером конца файла. Когда файл открывается, ему ставится в соответствие поток. Потоки обеспечивают каналы передачи данных между файлами и программами. В начале выполнения программы автоматически открываются три файла и связанные с ними потоки ? это стандартный ввод, стандартный вывод и стандартная ошибка. Этими потоками можно управлять. Например, если программа имеет имя myprogr.exe, то запуск в командной строке >myprog.exe >result.txt перенаправит вывод (по умолчанию, он ориентирован на экран) в файл result.txt. Пример. Запись в файл (как принято в С). #include <stdio.h> FILE *f, *fopen(); void main() { int i,x; f=fopen("c:\\tmp\\result.txt","w"); for(i=0;i<100;i++) { x=i*2; fprintf(f," %d ",x); } fclose(f); } Режимы открытия файлов r ? открыть файл для чтения; w ? создать файл для записи; a ? добавление в конец файла (или создание, если файл не обнаружен); r+ ? открытие файла для обновления (чтение и запись) Чтение из файла Можно использовать функцию FILE *fp; int n; float x; ?. fscanf(fp,?%d?,&n); fscanf(fp,?%f?,&x); Пример. Создаём файл (test.txt) для записи, записываем в него информацию и закрываем. Снова открываем этот же файл, но уже для чтения, выводим часть информации из файла на экран. #include <iostream> #include <fstream> using namespace std; int main() { ofstream fout("test.txt"); // создание файла для вывода if(!fout) { cout << "Файл открыть невозможно\n"; return 1; } fout << "Hello!\n"; fout << 100 <<' '<<200<< endl; fout.close(); ifstream fin("test.txt"); // открытие файла для ввода if(!fin) { cout << "Файл открыть невозможно\n"; return 1; } char str[80]; int i; fin >> str >> i; cout << str << ' ' << i << endl; fin.close(); return 0; } Пример. Запись в файл матрицы чисел с плавающей точкой. Числа создаются с помощью генератора случайных чисел. Запись в файл производится построчно ? каждая строка матрицы с новой строки. /* Запись матрицы в файл. Каждая строка матрицы - перевод на новую строку в файле */ // FILE *fp; char * fileName="d:\\temp\\resAA.txt"; fp=fopen(fileName,"w"); // заполнение массива числами const int n=4; double a[n][n]; for (int i=0;i<n;i++) for (int j=0;j<n;j++) a[i][j]= ((rand() % 10) +i*j)*0.1; // запись в файл for (int i=0;i<n;i++) { for (int j=0;j<n;j++) fprintf(fp,"%f ",a[i][j]); // или fprintf(fp,"%e ",a[i][j]); fprintf(fp,"\n"); } fclose(fp);

лабораторная работа (4 часа(ов)):

Упражнения на использование файлов. Пример. Чтение данных из файла в массив. Количество данных неизвестно. В программе файл открывается дважды: первый раз, чтобы подсчитать количество элементов, а второй раз ? для считывания данных в динамический массив. FILE *fp; char * fileName="d:\\temp\\dannn.txt"; fp=fopen(fileName,"r"); // узнаем сколько чисел в файле int r,n=0; while (!feof(fp)) // пока не конец файла { fscanf(fp,"%d",&r); n++; } fclose(fp); // закрываем файл int *mx= new int [n]; // динамический массив из n элементов fp=fopen(fileName,"r"); // снова открываем // чтение из файла for (int i=0;i<n;i++) { fscanf(fp,"%d",&r); mx[i]=r; } fclose(fp); // печать: for (int i=0;i<n;i++) printf("%d ",mx[i]); Пример (Файловый ввод в С++). Чтение данных из файла в массив. Количество данных неизвестно. Ранее, эта операция была выполнена с помощью функций fscanf() и feof(). Файл открывается дважды: сначала для вычисления количество элементов в файле, а затем ? для считывания данных в динамический массив. char * fileName="d:\\temp\\dannn.txt"; ifstream fin(fileName); // открытие файла // узнаем сколько чисел в файле int r,n=0; while (!fin.eof()) // пока не конец файла { fin>>r; n++; } fin.close(); // закрываем файл // int *mx= new int [n]; // динамический массив ifstream fin2(fileName); // снова открываем файл // чтение из файла for (int i=0;i<n;i++) fin2>>mx[i]; fin2.close(); // закрываем файл // печать: for (int i=0;i<n;i++) cout<<mx[i]<<" ";

Тема 21. Структуры.

лекционное занятие (2 часа(ов)):

Структура ? тип данных, определяемый пользователем. struct Cdrom{ int speed;//скорость int year;// год выпуска }; 2. Пример. В этом примере для доступа к полям структуры используются функции?члены (методы) #include <iostream> using namespace std; struct Cdrom{ int speed;//скорость int year;// год выпуска void set_speed(int s){speed=s;} int get_speed(){return speed;} void set_year(int y){year=y;} int get_year(){return year;} }; int main() { Cdrom acer; acer.set_speed(40); acer.set_year(2006);//доступ к элементам структуры Cdrom *lg=new Cdrom; lg->set_speed(48); (*lg).set_year(2007); cout<<"\nCd-Rom Acer "<<acer.get_speed(); cout<<" god vypuska "<<acer.get_year(); cout<<"\nCd-Rom LG "<<(*lg).get_speed(); cout<<" god vypuska "<<lg->get_year(); return 0; }

лабораторная работа (2 часа(ов)):

Упражнения на использование структур.

Тема 22. Алгоритмы приближенного решения уравнений.

лекционное занятие (2 часа(ов)):

Численное решение нелинейных уравнений. Реализация алгоритма простой итерации, алгоритма Ньютона, половинного деления.

лабораторная работа (2 часа(ов)):

Реализация методов.

Тема 23. Алгоритмы работы с матрицами.

лекционное занятие (4 часа(ов)):

Матричная арифметика.

лабораторная работа (4 часа(ов)):

Реализация методов.

Тема 24. Алгоритмы решения систем линейных уравнений. Реализация метода исключения Гаусса. Вычисление определителей. Вычисление обратной матрицы.

лекционное занятие (4 часа(ов)):

Наиболее распространенным методом решения систем линейных алгебраических уравнений является метод последовательного исключения неизвестных. Обычно этот метод называют методом исключения Гаусса. Алгоритм решения состоит из двух этапов: прямого хода и обратной подстановки. Решение методом Гаусса с выбором ведущего элемента. Отсылаем к книге Форсайт Дж., Малькольм М., Моулер К. Машинные методы математических вычислений: Пер. с англ. ? М.: Мир, 1980. В этой книге приведен пример (стр. 46), иллюстрирующий важность операции выбора ведущего элемента в методе исключения Гаусса. Метод исключения можно применить для вычисления определителя матрицы. Достаточно выполнить прямой ход метода, убрав из него операции со столбцом свободных членов и вычислить знак определителя. Прямой ход состоит из операций перестановки строк, деления строк на их ведущие элементы и вычитания строк. Операция вычитания из одной строки матрицы линейной комбинации других строк не изменяет определителя матрицы. При делении строки матрицы на число определитель также делится на это число. Перестановка любых строк матрицы меняет знак определителя. После прямого хода метода исключений матрица приводится к треугольному виду. Для любой невырожденной матрицы A существует единственная матрица B такая, что A•B=B?A=E, где E ? единичная матрица, т.е. Для нахождения элементов обратной матрицы необходимо решить систем линейных уравнений с одинаковой матрицей , но с различными правыми частями. Прямой ход метода исключений проводится только один раз. Затем требуется раз выполнить обратный ход, предварительно пересчитав правые части, учитывая, при этом, перестановки, связанные с выбором ведущих элементов.

лабораторная работа (6 часа(ов)):

Реализация методов. Пример. #include "stdafx.h" #include <iostream> #include <cmath> using namespace std; int inverse(double **, //Исходная матрица double **, // Обратная к исходной int // порядок матрицы); int _tmain(int argc, _TCHAR* argv[]) { double **a, **b; int n=4; // Порядок матриц int i,j; // Выделяем память для исходной матрицы: a=new double* [n]; for (i=0;i<n;i++) a[i]=new double [n]; // Выделяем память для обратной матрицы: b=new double* [n]; for (i=0;i<n;i++) b[i]=new double [n]; // заполнение исходной матрицы: a[0][0]=1.8;a[0][1]=-3.8;a[0][2]=0.7; a[0][3]=-3.7; a[1][0]=0.7;a[1][1]=2.1; a[1][2]=-2.6;a[1][3]=-2.8; a[2][0]=7.3;a[2][1]=8.1; a[2][2]=1.7; a[2][3]=-4.9; a[3][0]=1.9;a[3][1]=-4.3;a[3][2]=-4.9;a[3][3]=-4.7; // Вычисление обратной матрицы: if (!inverse((double **)a,(double **)b,n)) { // Печатаем элементы матрицы cout<<"\n A^(-1)=\n"; for (i=0;i<n;i++) { for (j=0;j<n;j++) cout<<b[i][j]<<" "; cout<<"\n"; } } return 0; } int inverse(double **a, //Исходная матрица double **y, // Обратная к исходной int n // порядок матрицы) { // Вычисление обратной матрицы. // double max_k, t; int i,j,k,m; if (n==1) // Система имеет порядок =1 { if (fabs(a[0][0])>0) { y[0][0]=1./a[0][0]; return 0; } else return 1; } // Порядок системы >1 // Начальные значения обратной матрицы: for (i=0;i<n;i++) { for (j=0;j<n;j++) y[i][j] =0.; y[i][i] = 1.; } // Прямой ход for (k=0;k<n-1;k++) {//2 //Найдем ведущий элемент в k-ом столбце: max_k = a[k][k]; m = k; for (i=k+1;i<n;i++) if (fabs(max_k)<fabs(a[i][k])) { m=i; max_k=a[m][k]; } //Ведущий элемент расположен в m-ой строке. //Система не разрешима, если этот элемент нулевой if (fabs(max_k)<0.00000001) return m; // Ненулевой ведущий элемент: if (m!=k) // перестановка строк m и k: { // 4 for (j=k;j<n;j++) { t=a[m][j]; a[m][j]=a[k][j]; a[k][j]=t; } // Перестановка в массиве y: for (j=0;j<n;j++) { t=y[m][j]; y[m][j]=y[k][j]; y[k][j]=t; } } // 4 // Делим k-ю строку на max_k: for (j=k+1;j<n;j++) a[k][j] /= max_k; for (j=0;j<n;j++) y[k][j] /= max_k; // Исключение по столбцам: for (i=k+1;i<n;i++) { for (j=k+1;j<n;j++) a[i][j] -= a[k][j] * a[i][k]; for (j=0;j<n;j++) y[i][j] -= y[k][j] * a[i][k]; } }// 2 // Вычисление элементов обратной матрицы // Для каждого k=0,...,n-1 проводим обратный ход, // используя в качестве // правых частей системы k-й столбец массива y for (k=0;k<n;k++) { // 1_ if (fabs(a[n-1][n-1])>0) { // 2_ y[n-1][k] /= a[n-1][n-1]; for (i=n-2;i>=0;i--) { // 3_ t=0; for (j=i+1;j<n;j++) t += a[i][j] * y[j][k]; y[i][k] -= t; } // 3_ } // 2_ } // 1_ return 0; } //inverse()

Тема 25. Повышение точности вычислений.

лекционное занятие (2 часа(ов)):

Вычисления с двойной точностью.

лабораторная работа (2 часа(ов)):

Практические примеры.

Тема 26. Основные понятия объектно-ориентированного программирования (ООП).

лекционное занятие (2 часа(ов)):

Объектно-ориентированное программирование В классе объединены данные и функции управления этими данными. Такое объединение называется инкапсуляцией. Размещение данных в разделе private называют сокрытием данных.

лабораторная работа (2 часа(ов)):

Практические примеры.

Тема 27. Классы. Атрибуты и операции. Реализация операций (функций-членов) класса вне класса. Инкапсуляция. Модификаторы доступа private и public.

лекционное занятие (4 часа(ов)):

Реализация функций-членов вне класса. Используется операция ?::? (операция доступа к компоненте класса) #include <iostream> using namespace std; class Cdrom{ int speed;//скорость int year;// год выпуска public: void set_speed(int s); int get_speed(); void set_year(int y); int get_year(); }; void Cdrom::set_speed(int s) { speed=s; } inline int Cdrom::get_speed() { return speed; } void Cdrom::set_year(int y) { year=y; } int Cdrom::get_year(){ return year; } int main() { Cdrom acer; acer.set_speed(40); acer.set_year(2006); cout<<"\nCd-Rom Acer "<<acer.get_speed(); cout<<" god vypuska "<<acer.get_year(); return 0; }

лабораторная работа (4 часа(ов)):

Практические примеры.

Тема 28. Конструкторы и деструкторы. Конструктор по умолчанию. Конструктор копирования.

лекционное занятие (4 часа(ов)):

Конструктор вызывается каждый раз, когда создаётся объект данного типа; деструктор ? при уничтожении. При преобразованиях типов с участием экземпляров классов тоже вызываются конструкторы и деструкторы. Пример. Конструктор класса. Имя конструктора совпадает с именем класса. #include <iostream> using namespace std; class Cdrom{ int speed; int year; public: Cdrom(int s,int y){speed=s;year=y;} void set_speed(int s) {speed=s;} int get_speed() {return speed;} void set_year(int y){year=y;} int get_year(){return year;} }; int main() { Cdrom acer(40,2006); Cdrom sony;// Это уже ошибка cout<<"\nCd-Rom Acer "<<acer.get_speed(); cout<<" god vypuska "<<acer.get_year(); return 0; } Создание объекта в C++ всегда сопровождается вызовом конструктора. Если класс совсем не содержит конструкторов (как в примере 1), то система создаёт конструктор по умолчанию. Cdrom acer; ? это вызов конструктора по умолчанию. Нужно запомнить, что компилятор создает конструктор, заданный по умолчанию, только пока в классе нет конструкторов. После включения в класс конструктора, компилятор уже не создаст конструктор по умолчанию и это уже задача программиста. В примере 2 можем создавать объекты только так Cdrom acer(40,2006); т.е. указывая аргументы. В этом примере уже не можем создать объект, просто указав его имя Cdrom sony; Можно иметь только один конструктор по умолчанию. Конструкторов может быть несколько, и они создаются с учётом правил перегрузки функций. Каждый класс имеет неявно объявленные конструктор без параметров ? конструктор по умолчанию, копирующий конструктор, копирующий оператор присваивания и деструктор. Класс может иметь несколько конструкторов (с разными наборами параметров), но только один деструктор, причём у него нет параметров. Деструктор Деструктор только один и имеет такое же имя, как и класс, только перед именем ставим символ ?~?. Деструктор не имеет аргументов.

лабораторная работа (4 часа(ов)):

Практические примеры.

Тема 29. Объекты. Массив объектов. Инициализация массива объектов. динамическое выделение памяти объектам. Указатели на объекты. Указатель this. Передача объектов функции по ссылке и по значению.Дружественные функции. Возвращение объекта из функции.

лекционное занятие (4 часа(ов)):

#include <iostream> using namespace std; class Cdrom{// как в предыдущем примере: int speed;//скорость int year;// год выпуска public: Cdrom(){speed=32;year=2005;cout<<"\nConstr-1";} Cdrom(int s,int y=2006){speed=s;year=y;cout<<"\nConstr-2";} ~Cdrom(){cout<<"\nDestructor";} void set_speed(int s) {speed=s;} int get_speed() {return speed;} void set_year(int y){year=y;} int get_year(){return year;} }; int main() { Cdrom acer(40,2006); // Объект acer Cdrom sony; // Объект sony Cdrom lg(52); // Объект lg cout<<"\nCd-Rom Acer "<<acer.get_speed(); cout<<" god vypuska "<<acer.get_year(); cout<<"\nCd-Rom Sony "<<sony.get_speed(); cout<<" god vypuska "<<sony.get_year(); cout<<"\nCd-Rom LG "<<lg.get_speed(); cout<<" god vypuska "<<lg.get_year(); return 0; } Пример. Инициализация массива объектов с помощью конструктора с двумя параметрами. #include "stdafx.h" #include <iostream> using namespace std; class Cdrom{ int speed;//скорость int year;// год выпуска public: Cdrom(){speed=32;year=2005;cout<<"\nConstr-1";} Cdrom(int s,int y=2006){speed=s;year=y;cout<<"\nConstr-2";} ~Cdrom(){cout<<"\nDestructor";} void set_speed(int s) {speed=s;} int get_speed() {return speed;} void set_year(int y){year=y;} int get_year(){return year;} }; int _tmain(int argc, _TCHAR* argv[]) { Cdrom ob[4]= {Cdrom(40,2006),Cdrom(42,2006),Cdrom(44,2007),Cdrom(46,2008)}; for (int i=0;i<4;i++) { cout<<ob[i].get_speed()<<"\n"; cout<<ob[i].get_year()<<"\n"; } return 0; } Дружественные функции Пример. Дружественные функции. Функция sravnim() возвращает положительное число, если легковая машина (car) быстрее грузовика (truck). Возвращает 0 при одинаковых скоростях. Возвращает отрицательное число, если грузовик быстрее легковой машины. #include <iostream> using namespace std; class truck; // предварительное объявление class car { int passengers; int speed; public: car(int p, int s) { passengers = p; speed = s; } friend int sravnim(car c, truck t); }; class truck { int weight; int speed; public: truck(int w, int s) { weight = w; speed = s; } friend int sravnim(car c, truck t); }; int sravnim(car c, truck t) { return c.speed - t.speed; } int main() { int t; car c1(6, 55), c2 (2, 120); truck t1(10000, 55), t2(20000, 72); cout << "Сравнение значений c1 и t1:\n"; t = sravnim(c1, t1); if(t<0) cout << "Грузовик быстрее. \n"; else if(t==0) cout << "Скорости машин одинаковы. \n"; else cout << "Легковая машина быстрее. \n"; cout << "\nСравнение значений c2 и t2:\n"; t = sravnim(c2, t2); if(t<0) cout << "Грузовик быстрее. \n"; else if(t==0) cout << "Скорости машин одинаковы. \n"; else cout << "Легковая машина быстрее. \n"; return 0; }

лабораторная работа (4 часа(ов)):

Практические примеры. Пример. Передача объектов функции. Передача по значению. В функции создается копия объекта. Изменение параметра внутри функции не влияет на объект, используемый в вызове. #include "stdafx.h" #include <iostream> using namespace std; class Cdrom{ int speed;//скорость int year;// год выпуска public: Cdrom(){speed=32;year=2005;cout<<"\nConstr-1";} Cdrom(int s,int y=2006){speed=s;year=y;cout<<"\nConstr-2";} ~Cdrom(){cout<<"\nDestructor";} void set_speed(int s); int get_speed() {return speed;} void set_year(int y){year=y;} int get_year(){return year;} }; void Cdrom::set_speed(int s) { speed=s; } void model(Cdrom v) { if (v.get_speed() <40) cout<<"\n Slow cdrom-model\n"; else if (v.get_speed() <50) cout<<"\nNomal cdrom-model\n"; else cout<<"\nFast cdrom-model\n"; } int _tmain(int argc, _TCHAR* argv[]) { Cdrom cdr(40,2005); model(cdr); return 0; } Пример. Передача объектов функции. Теперь при вызове функции передается адрес объекта и функция может изменить значение аргумента, адрес которого используется при вызове. #include "stdafx.h" #include <iostream> using namespace std; class Cdrom{ int speed;//скорость int year;// год выпуска public: Cdrom(){speed=32;year=2005;cout<<"\nConstr-1";} Cdrom(int s,int y=2006){speed=s;year=y;cout<<"\nConstr-2";} ~Cdrom(){cout<<"\nDestructor";} void set_speed(int s); int get_speed() {return speed;} void set_year(int y){year=y;} int get_year(){return year;} }; void Cdrom::set_speed(int s) { speed=s; } void model(Cdrom *v) { if (v->get_speed() <40) cout<<"\n Slow cdrom-model\n"; else if (v->get_speed() <50) cout<<"\nNomal cdrom-model\n"; else cout<<"\nFast cdrom-model\n"; v->set_year(2009); } int _tmain(int argc, _TCHAR* argv[]) { Cdrom cdr(40,2005); model(&cdr); return 0; } Пример. Возвращение объекта из функции #include "stdafx.h" #include <iostream> using namespace std; class Cdrom{ int speed;//скорость int year;// год выпуска public: Cdrom(){speed=32;year=2005;cout<<"\nConstr-1";} Cdrom(int s,int y=2006){speed=s;year=y;cout<<"\nConstr-2";} ~Cdrom(){cout<<"\nDestructor";} void set_speed(int s); int get_speed() {return speed;} void set_year(int y){year=y;} int get_year(){return year;} }; void Cdrom::set_speed(int s) { speed=s; } void model(Cdrom &v) { if (v.get_speed() <40) cout<<"\n Slow cdrom-model\n"; else if (v.get_speed() <50) cout<<"\nNomal cdrom-model\n"; else cout<<"\nFast cdrom-model\n"; } Cdrom inputcd() { int sp;//скорость int ye;// год выпуска Cdrom cdr; cout<<"\n Speed= "; cin>>sp; cdr.set_speed(sp); cout<<"\n Year= "; cin>>ye; cdr.set_year(ye); return cdr; } int _tmain(int argc, _TCHAR* argv[]) { Cdrom cdrm; cdrm = inputcd(); model(cdrm); return 0; }

Тема 30. . Наследование. Правила наследования с модификаторами private и public. Модификатор доступа protected. Конструкторы и наследование. Вызов методов Множественное наследование.базового класса в производном.

лекционное занятие (6 часа(ов)):

Насле́дование ? один из четырёх важнейших механизмов объектно-ориентированного программирования (наряду с инкапсуляцией, полиморфизмом и абстракцией), позволяющий описать новый класс на основе уже существующего (родительского), при этом свойства и функциональность родительского класса заимствуются новым классом. Класс, от которого произошло наследование, называется базовым или родительским (англ. base class). Классы, которые произошли от базового, называются потомками, наследниками или производными классами (англ. derived class). Наследование в C++: class A{ //базовый класс }; class B : public A{ //public наследование } class C : protected A{ //protected наследование } class Z : private A{ //private наследование } Спецификатор доступа protected действует как private, но защищенные (protected) члены базового класса доступны для членов всех производных классов этого класса. Когда базовый класс наследуется производным классом как открытый (public), защищенный (protected) член базового класса остается защищенным (protected) членом производного класса (т. е. снова protected). Когда базовый класс наследуется как закрытый (private), то защищенный (protected) член базового класса становится закрытым (private) членом производного класса. Базовый класс может наследоваться с доступом protected. В этом случае открытые (public) и защищенные (protected) члены базового класса становятся защищенными (protected) членами производного класса, а закрытые (private) так и остаются закрытыми. Диаграмма классов. Правила области действия и разрешение имен при наследовании. Перегрузка и сокрытие имен. Вызов метода базового класса, скрытого производным классом. Прямое и косвенное наследование. В C++ производный класс может иметь более одного базового класса. При множественном наследовании производный класс наследует элементы данных всех базовых классов и все методы этих классов. Область памяти, которую занимает объект производного класса, представляет собой сумму пространства, занимаемого в памяти объектами базовых классов. Правила доступа для множественного наследования те же, что и для простого наследования. Все конструкторы базового класса вызываются до вызова конструкторов производного класса. Располагаются они в том порядке, в котором базовые классы перечислены в объявлении производного класса. Если в двух базовых классах совпадают имена элементов данных или методов, то объект производного класса содержит обе копии. C++ не имеет правил предшествования для доступа к данным и методам. Неоднозначность разрешается использованием явной квалификации. Иерархия классов при множественном наследовании представляет собой граф, в простом наследовании ? дерево.

лабораторная работа (6 часа(ов)):

Практические примеры.

Тема 31. Виртуальные методы. Статическое и динамическое связывание.

лекционное занятие (4 часа(ов)):

Указатель базового класса может указывать на объект любого класса, производного от этого базового класса. Обратный порядок недействителен. В C++ метод производного класса может иметь иную реализацию (?поведение?), чем метод (с тем же именем и сигнатурой) базового класса и поведение метода будет зависеть от вызывающего объекта. Такое поведение называется полиморфным (?имеющим много форм?). Полиморфное наследование можно реализовать двумя способами: 1. Переопределением методов базового класса в производном классе 2. Использованием виртуальных методов Интерпретация вызова функции в исходном коде при выполнении определенного его блока называется связыванием имени функции. Связывание, происходящее во время компиляции, называется статическим (или ранним) связыванием. При динамическом (или позднем) связывании, компилятор генерирует код, позволяющий выбирать нужный виртуальный метод во время работы программы. Компиляторы при обработке виртуальных функций добавляют к каждому объекту скрытый элемент, который содержит указатель на массив адресов функций. Этот массив называется таблицей виртуальных методов (vtbl) и содержит адреса виртуальных функций, объявленных для объектов этого класса. Когда вызывается виртуальная функция, программа просматривает адрес таблицы, сохраненный в объекте, и обращается к таблице адресов функций. Несколько правил использования виртуальных методов Если объявление метода класса в базовом классе начинается с ключевого слова virtual, то этот метод становится виртуальным для базового класса и всех производных классов. Ключевое слово virtual указывается только в спецификации базового класса, не требуется повторять его в спецификации производного класса. Если виртуальный метод вызывается с помощью ссылки на объект или указателя на объект (->), в программе реализуется динамическое связывание и будет использоваться метод, определенный для типа объекта, а не для типа указателя или ссылки. Transport* ppp; // Transport* - тип указателя ppp= new Velo("Sputnik",2); // Velo ? тип объекта ppp->PlaySignal(); // PlaySignal() из Velo Конструкторы не могут быть виртуальными. Деструкторы должны быть виртуальными. Когда базовый указатель ссылается на объект производного класса, вызывается деструктор базового класса. Если управление памятью динамически распределяемой области памяти осуществляется в производном классе, она не будет возвращена базовым деструктором, в результате произойдет утечка памяти. Поэтому деструктор базового класса следует объявлять виртуальным.

лабораторная работа (4 часа(ов)):

Практические примеры.

Тема 32. Абстрактные классы.

лекционное занятие (2 часа(ов)):

Классы, объекты которых не создаются, называются абстрактными. Такой класс должен включать метод, не содержащий реализации. Этот метод должен переопределяться в производных классах. Нереализованные методы класса создаются с помощью механизма чисто виртуальных функций. Виртуальная функция объявляется как чисто виртуальная, если в прототипе этой функции указывается ?=0? в самом конце. virtual void Draw()=0; Класс с одной или несколькими чисто виртуальными функциями является (называется) абстрактным. Иерархия классов ? средство последовательного построения классов. Иерархия абстрактных классов предоставляет способ выражения концепций, не загроможденных вопросами реализаций.

лабораторная работа (2 часа(ов)):

Практические примеры.

Тема 33. Константные переменные. Указатель на константу. Константные параметры функций.

лекционное занятие (2 часа(ов)):

Ключевое слово const. Разный смысл ключевого слова const. Константные переменные Если переменная объявлена с ключевым словом const, в программе не должно быть операторов, изменяющих значение этой переменной. const double pi=3.1415; // нельзя изменять pi // pi=3.14156; // ошибка Объявление константной переменной должно сопровождаться инициализацией переменной. Замечание. В языке С константы определялись с помощью директивы #define PI 3.1415 Указатель на константу (const Тип *) Адрес может меняться, а содержимое памяти по этому адресу ? нет. double s=2; double * u=&s; const double *t=&s; *u=5; // так можно s=100; // так можно //*t=6;// а так нельзя - const запрещает изменять значение *t double yy=3; t=&yy; // так можно Константный указатель (Тип* const) Адрес изменять нельзя, но можно изменять содержимое памяти по этому адресу. double x=3, v=5; double * const pt = &x; // pt указывает только на x cout<<"\n *pt= "<< *pt<<" x= "<<x<<endl; *pt=11; //pt=&v; // этот оператор приведёт к ошибке Константный указатель на константу (const Тип* const) Константный указатель на константу изменять нельзя. // 4. Константный указатель на константу double r =3.14156; double rr =3.14; const double * const pr=&r; cout<<"\n r="<<r<<" *pr= "<<*pr<<endl; *pr = 3.14; // Ошибка - изменять значение нельзя pr = &rr; // Ошибка - изменять адрес нельзя Константные параметры функций Если формальный аргумент объявлен неконстантным, то и фактический аргумент, используемый при вызове, обязан быть неконстантным. Если формальный аргумент объявлен константным, то фактический может быть неконстантным.

лабораторная работа (2 часа(ов)):

Практические примеры.

Тема 34. Операции с файлами в С++ (классы ofstream и ifstream). Чтение из файла, запись в файл.

лекционное занятие (2 часа(ов)):

Рекомендуется в программах на С++ использовать ввод-вывод с использованием набора классов, определенных в заголовочных файлах iostream (iostream.h) и fstream (fstream.h). С точки зрения программы, написанной на С++, ввод-вывод представляется как поток байтов. При вводе программа читает байты из потока ввода, а при выводе вставляет байты в поток вывода. Байты потока поступают с клавиатуры, из другой программы, с устройства хранения данных (например, диск). Управление вводом предполагает две стадии: ? Присоединение потока ввода к программе ? Связывание потока с файлом.

лабораторная работа (2 часа(ов)):

Практические примеры. Стандартные задачи на файлы 1. Дан текстовый файл f, получить копию файла f в файле g. 2. Даны текстовые файлы f и g. Записать в файл h сначала компоненты файла f, а затем ? компоненты файла g. 3. Даны текстовые файлы f и g. Написать программу, которая печатает информацию вперемежку из двух файлов: одна строка из файла f, другая ? из файла g и т.д. 4. Написать программу сравнения двух текстовых файлов ? на печать должна выводится первая из различающихся строк, а также номер этой строки и позицию символа, в котором они различаются. 5. Дан текстовый файл f. Создать файл g, образованный из файла f заменой всех прописных букв одноименными строчными. 6. Дан текстовый файл f. Подсчитать количество строк и символов в файле f. Указание (из А.Богатырев. Язык Си в системе UNIX, стр. 130): надо подсчитать количество символов ?\n? в файле и учесть, что последняя строка файла может не иметь этого символа на конце. Поэтому, если последний символ файла не есть ?\n?, то нужно добавить к счетчику строк 1. 7. Дан текстовый файл f. Подсчитать количество вхождений каждого из символов алфавита в файле f. 8. Дан текстовый файл f. Подсчитать количество слов в файле f. 9. Дан текстовый файл f. Подсчитать в файле f количество слов заданной длины n. 10. Подсчитать количество вхождений данного слова s в текстовый файл. 11. Дан текстовый файл f, содержащий информацию в кодировке DOS (OEM ? 866). Преобразовать информацию в кодировку Windows (Win-1251). 12. Дан текстовый файл f, содержащий информацию на русском языке. Преобразовать текст, заменив символы кириллицы на ?подходящие? латинские буквы (для букв, не имеющих аналогов в латинице, использовать сочетания букв, например, ?ж? ? ?zh?, ?х? ? ?kh?, ?ц? ? ?ts?, ?ч? ? ?ch?, ?ш? ? ?sh?, ?щ? ? ?tsh?, ?ю? ? ?ju?, ?я? ? ?ja?). Преобразованный текст записать в файл g. 13. Дан текстовый файл f. Найти слово (слова), встречающиеся наиболее часто. 14. Дан текстовый файл f. Вычислить количество вхождений каждого символа в файле f. Результат сохранить в массиве.

Тема 35. Перегрузка операций.

лекционное занятие (2 часа(ов)):

Перегрузка операций Это свойство языка позволяет использовать привычные символы операций, такие как, +, -, *, не только для числовых типов (например, int, double), но и для объектов. Перегрузка операторов является одним из видов перегрузки функций. Для перегрузки оператора создается оператор-функция. Основная форма оператор-функции возвращаемый-тип operatorOP(список-аргументов) где OP ? символ операции, которая должна быть перегружена, т.е. operator+() перегружает операцию +, а operator*() перегружает операцию *. Ограничения на перегрузку 1. Перегруженная операция должна иметь хотя бы один операнд с типом, определенным пользователем ? это необходимо, чтобы запретить перегрузку операций для стандартных типов (таких как, int) данных. 2. Нельзя менять приоритет операций. 3. Нельзя менять число операндов операции. 4. Невозможно создавать новые символы операций.

лабораторная работа (2 часа(ов)):

Практические примеры.

Тема 36. Раздельная компиляция, "стражи включения".

лекционное занятие (2 часа(ов)):

Раздельная компиляция применяется для разделения программы на несколько файлов с возможностью отдельной компиляции каждого из них и объединением в одно целое (например, exe?файл) на заключительном этапе (подробнее, см., напр., Прата С., Язык программирования C++, Стауструп Б. Язык программирования C++). Заголовочный файл следует включать в проект только один раз. Если какие-то заголовочные файлы включают (с помощью #include) другие заголовочные файлы, то это правило трудно проконтролировать. Стандартная методика предотвращения многократных включений заголовочных файлов, основана на использовании стражей включения ? директив препроцессора #ifndef и #endif.

лабораторная работа (2 часа(ов)):

Практические примеры.

Тема 37. Пространства имен.

лекционное занятие (2 часа(ов)):

Технология пространства имён (Namespace) используется в большинстве современных языков программирования. Пространство имён группирует некоторое множество идентификаторов и используется как средство исключения конфликта имён. Пространства имён можно рассматривать как механизм отражения логического группирования (см., напр., Стауструп Б. Язык программирования C++). Пространства имён определяются либо на глобальном уровне, либо внутри других пространств имён. Нельзя создавать пространства имён в блоке. Имя, объявленное в пространстве имён, обладает внешним связыванием.

лабораторная работа (2 часа(ов)):

Практические примеры.

Тема 38. Структуры данных. Внутренняя и внешняя сортировки

лекционное занятие (2 часа(ов)):

Под термином сортировка понимается процедура перестановки элементов множества в определённом порядке. Методы сортировки классифицируются на внутренние и внешние. При внутренней сортировке данные размещаются в оперативной памяти. При внешней сортировке данные размещаются во внешней памяти. К внешней сортировке прибегают в случаях, когда невозможно разместить в оперативной памяти все данные.

Тема 39. Сортировка числового массива методом пузырька. Сортировка массива объектов методом пузырька. Сортировка числового массива методом просеивания. Сортировка числового массива методом Шелла. Сортировка числового массива методом быстрой сортировки.

лекционное занятие (6 часа(ов)):

Метод пузырька. Производятся последовательные просмотры массива и каждый раз пару за парой сравниваются соседние числа. Если числа в паре не расположены в порядке возрастания меняем их местами. Затем переходим к следующей паре. Сортировка считается законченной, если в ходе просмотра не была произведена ни одна перестановка. Можно использовать переменную, например, flag, чтобы зафиксировать факт перестановки, ? в начале каждого просмотра переменной присваивается значение 0 (False) и если в ходе просмотра была выполнена хотя бы одна перестановка, значение переменной flag меняется на 1 (True). Таким образом, по значению этой переменной определяем, нужен или нет ещё один просмотр. Метод просеивания Выполняется так же как метод пузырька, но после перестановки элементов величина с меньшим значением передвигается к началу массива, насколько это возможно. Она сравнивается в обратном порядке со всеми предшествующими элементами массива. Если значение меньше, чем у предшествующего элемента массива, то выполняется перестановка. Как только встречается элемент с меньшим значением, то процесс продвижения к началу массива прекращается и нисходящее сравнение возобновляется с той же позиции, с которой начался обратный ход. void sort_sift(int *x, int dim, int (*sravn)(int, int)) { int i,j,t; for (i=0;i<dim;i++){ t=x[i]; for (j=i-1;(j>=0) && (sravn(x[j],t)>0); j--) x[j+1]=x[j]; x[j+1]=t; } } Метод Шелла Метод предложен в 1959 году, автор ? Donald Shell. Так же как и метод просеивания, состоит из прямого и обратного хода. Но сравниваются и обмениваются не непосредственные соседи, а элементы, отстоящие на заданном расстоянии. Когда обнаружена перестановка, цепочка вторичных сравнений охватывает те элементы, которые входили в последовательность первичных просмотров. void sort_shell(int *x, int dim, int (*sravn)(int, int)) { int i,j,d,t; d=dim/2; while (d>=1){ for (i=d;i<dim;i++){ t=x[i]; for (j=i-d;(j>=0) && (sravn(x[j],t)>0); j-=d) x[j+d]=x[j]; x[j+d]=t; } d= d/2; } } Быстрая сортировка 1. Выбираем в массиве (случайным образом) какой-нибудь элемент . 2. Просматриваем массив, двигаясь слева направо, пока не найдем элемент . 3. Просматриваем список, двигаясь справо налево, пока не найдем элемент . 4. Меняем местами элементы и . 5. Продолжим процесс просмотра, пока два просмотра не встретятся. В результате массив разделится на две части: левую с ключами, меньшими чем , и правую ? с ключами, большими . 6. Применяем приведённую процедуру для каждой из полученных частей.

практическое занятие (6 часа(ов)):

Практические примеры. // sort-08.cpp : Быстрая сортировка // #include "stdafx.h" #include <iostream> using namespace std; void Swap(int *, int *); void quickSortR(int* , int); int _tmain(int argc, _TCHAR* argv[]) { const int n=20; int i; int x[n]={19,20,1,2,3,5,4,7,6,18,8,17,9,11, 10,12,14,13,16,15}; printf("\n"); for (i=0;i<n;i++) printf(" %d ",x[i]); quickSortR(x,n-1); printf("\n\n Sort Up:\n"); for (i=0;i<n;i++) printf(" %d ",x[i]); printf("\n"); return 0; } void quickSortR(int * a, int dim) { // На входе - массив a[], a[dim] - его последний элемент. int i = 0, j = dim; // поставить указатели на исходные места int temp, p; p = a[dim>>1]; // центральный элемент // процедура разделения do { while (a[i] < p) i++; while (a[j] > p) j--; if (i <= j) { Swap(&a[i],&a[j]); // перестановка i++; j--; } } while (i<=j); // рекурсивные вызовы, если есть, что сортировать if (j > 0) quickSortR(a, j); if (dim > i) quickSortR(a+i, dim-i); }

Тема 40. Метод естественного слияния.

лекционное занятие (2 часа(ов)):

Сортировка слиянием Как и quickSort, но, вместо разделения по опорному элементу массив просто делится пополам. Рекурсивный алгоритм обходит получившееся дерево слияния в прямом порядке. Каждый уровень представляет собой проход сортировки слияния - операцию, полностью переписывающую массив. Один из способов состоит в слиянии двух упорядоченных последовательностей при помощи вспомогательного буфера, равного по размеру общему количеству имеющихся в них элементов. Элементы последовательностей будут перемещаться в этот буфер по одному за шаг. Результатом является упорядоченная последовательность, находящаяся в буфере. Каждая операция слияния требует n пересылок и n сравнений, где n - общее число элементов.

практическое занятие (2 часа(ов)):

Практические примеры. // sort-merge.cpp : Сортировка слиянием // #include "stdafx.h" #include <iostream> using namespace std; void mergeSort(int numbers[], int , int); void merge(int a[], int, int, int); int _tmain(int argc, _TCHAR* argv[]) { const int n=20; int i; int x[n]={19,20,1,2,3,5,4,7,6,18,8,17,9,11, 10,12,14,13,16,15}; printf("\n"); for (i=0;i<n;i++) printf(" %d ",x[i]); // mergeSort(x, 0, n-1); // printf("\n\n Sort Up:\n"); for (i=0;i<n;i++) printf(" %d ",x[i]); printf("\n"); return 0; } void mergeSort(int a[], int lb, int ub) { int split; // индекс, по которому делим массив if (lb < ub) { // если есть более 1 элемента split = (lb + ub)/2; mergeSort(a, lb, split); // сортировать левую половину mergeSort(a, split+1, ub); // сортировать правую половину merge(a, lb, split, ub); // слить результаты в общий массив } } void merge(int a[], int lb, int split, int ub) { // Слияние упорядоченных частей массива в буфер temp // с дальнейшим переносом содержимого temp в a[lb]...a[ub] // текущая позиция чтения из первой последовательности a[lb]...a[split] int pos1=lb; // текущая позиция чтения из второй последовательности a[split+1]...a[ub] int pos2=split+1; // текущая позиция записи в temp int pos3=0; int *temp = new int[ub-lb+1]; // идет слияние, пока есть хоть один элемент в каждой последовательности while (pos1 <= split && pos2 <= ub) { if (a[pos1] < a[pos2]) temp[pos3++] = a[pos1++]; else temp[pos3++] = a[pos2++]; } // одна последовательность закончилась - // копировать остаток другой в конец буфера while (pos2 <= ub) // пока вторая последовательность непуста temp[pos3++] = a[pos2++]; while (pos1 <= split) // пока первая последовательность непуста temp[pos3++] = a[pos1++]; // скопировать буфер temp в a[lb]...a[ub] for (pos3 = 0; pos3 < ub-lb+1; pos3++) a[lb+pos3] = temp[pos3]; }

Тема 41. Алгоритмы внешней сортировки.

лекционное занятие (6 часа(ов)):

Если сортируемый файл настолько велик, что не помещается в оперативной памяти компьютера, то прибегают к методам сортировки, для которых используют термин внешняя сортировка (external sorting). Абстрактная модель сортировки построена на предположении, что сортируемый файл не помещается в оперативной памяти компьютера. В распоряжении имеется: N записей на внешнем устройстве, сортировку которых необходимо выполнить; объем оперативной памяти, достаточный для хранения M записей; 2P внешних устройств, которыми можно пользоваться во время сортировки. Большинство известных методов внешней сортировки основаны на принципах распределения и слияния. Во время первого прохода по файлу производится его разбиение на блоки (размером меньшим, чем объем оперативной памяти) и выполняются операции сортировки этих блоков. Затем отсортированные блоки сливают в новый файл. Метод естественного слияния Метод естественного слияния основывается на распознавании серий, их распределении и последующем слиянии. Сортировка выполняется за несколько шагов, в каждом из которых выполняются: ? распределение файла A по файлам B и C ? слияние файлов B и C в файл A. При распределении распознается первая серия элементов и переписывается в файл B, вторая серия записывается в файл C, следующая серия снова в B и т.д. При слиянии первая серия записей файла B сливается с первой серией файла C, вторая серия B со второй серией C и т.д. Если просмотр одного файла заканчивается раньше, чем просмотр другого (по причине разного числа серий), то остаток недопросмотренного файла целиком копируется в конец файла A. Процесс завершается, когда в файле A остается только одна серия.

практическое занятие (8 часа(ов)):

Практические примеры. Сортировка естественным слиянием. Вариант 1. Слияние файлов, а не серий! Листинг. // SortMergeFile2.cpp // #include "stdafx.h" #include <stdio.h> #include <iostream> using namespace std; FILE *fopen(); void MergeSort(char *fileA); void Distribute(char *fileA, char *fileB, char *fileC, int &sB,int &sC); void Merge(char *fileA, char *fileB, char *fileC); void show_file(char *fileA); int _tmain(int argc, _TCHAR* argv[]) { char fileA[]="c:\\tmp\\A.txt"; // Файл с данными show_file(fileA); MergeSort(fileA); printf("\n After Sorting \n"); show_file(fileA); return 0; } void MergeSort(char *fileA) { char fileB[]="B.txt"; // Вспомогательный файл char fileC[]="C.txt"; // Вспомогательный файл int sB, sC; sB=sC=2;// начальные значения (произвольно) // sB - число переходов с ленты B // sC - число переходов с ленты C while ((sB>=1) && (sC>=1)) // закончим, когда останется одна серия (переходов не будет) { // распределение по лентам (файлам): Distribute(fileA,fileB,fileC,sB,sC);// sB, sC - по ссылке // Слияние: Merge(fileA,fileB,fileC); }//1 }//MergeSort() void Distribute(char *fileA, char *fileB, char *fileC, int &sB,int &sC) { FILE *fA, *fB, *fC; int x, y; char wher; // распределение по лентам (файлам): fA=fopen(fileA,"r"); fB=fopen(fileB,"w"); fC=fopen(fileC,"w"); wher='B'; sB=sC=0; fscanf(fA,"%d",&x); fprintf(fB," %d ",x); while (!feof(fA))//пока не закончился файл fA { fscanf(fA,"%d",&y); if (y<x) // конец серии { switch (wher) { // меняем ленты (файлы) case 'B': wher='C'; sB++; break; case 'C': wher='B'; sC++; break; } }// if switch (wher) { // запись на ленты (файлы) case 'B': fprintf(fB," %d",y); break; case 'C': fprintf(fC," %d",y); break; } x=y; } fclose(fA); fclose(fB); fclose(fC); }//Distribute() void Merge(char *fileA, char *fileB, char *fileC) { FILE *fA, *fB, *fC; int x, y; // Слияние: fA=fopen(fileA,"w"); fB=fopen(fileB,"r"); fC=fopen(fileC,"r"); fscanf(fB,"%d",&x); fscanf(fC,"%d",&y); while ((!feof(fB)) && (!feof(fC))) /* пока не прочитаем файлы fB и fC */ { //3 if (x<=y) { fprintf(fA," %d",x); fscanf(fB,"%d",&x); } else { fprintf(fA," %d",y); fscanf(fC,"%d",&y); } } // 3- while while (!feof(fB)) // остаток файла fB { fprintf(fA," %d",x); fscanf(fB,"%d",&x); } while (!feof(fC)) // остаток файла fC { fprintf(fA," %d",y); fscanf(fC,"%d",&y); } fclose(fA); fclose(fB); fclose(fC); }//Merge() void show_file(char *fileA) { FILE *fA; int a; fA=fopen(fileA,"r"); printf("\n %s \n",fileA); while (!feof(fA)) { fscanf(fA,"%d",&a); printf("\n %d",a); } fclose(fA); }

Тема 42. Алгоритмы поиска

лекционное занятие (4 часа(ов)):

Алгоритм бинарного поиска.

практическое занятие (4 часа(ов)):

Практические примеры.

Тема 43. Деревья цифрового поиска

лекционное занятие (4 часа(ов)):

Алгоритмы построения деревьев поиска.

практическое занятие (4 часа(ов)):

Практические примеры.

Тема 44. Создание однонаправленного списка. Создание двунаправленного списка. Добавление нового узла в начало однонаправленного списка. Добавление нового узла в начало двунаправленного списка. Добавление нового узла в конец однонаправленного списка. Добавление нового узла в конец двунаправленного списка. Добавление нового узла в определенное место однонаправленного списка. Добавление нового узла в определенное место двунаправленного списка. Удаление первого узла однонаправленного списка. Удаление первого узла двунаправленного списка. Удаление последнего узла однонаправленного списка. Удаление последнего узла двунаправленного списка. Удаление заданного узла однонаправленного списка. Удаление заданного узла двунаправленного списка.

лекционное занятие (6 часа(ов)):

Линейный список - это конечная последовательность однотипных элементов (узлов), возможно, с повторениями. Количество элементов в последовательности называется длиной списка, причем длина в процессе работы программы может изменяться. При работе со списками чаще всего приходится выполнять следующие операции: ? найти элемент с заданным свойством; ? вставить дополнительный элемент до или после указанного узла; ? исключить определенный элемент из списка; ? упорядочить узлы линейного списка в определенном порядке. Методы хранения линейных списков разделяются на методы последовательного и связанного хранения. При последовательном хранении элементы линейного списка размещаются в массиве фиксированных размеров. Например, int a[10]; double z[20]; Элемент (узел) списка Элемент списка записывается в виде структуры struct ListInt{ int x; ListInt *next; }; или так typedef struct ListInt *Link; struct ListInt{ int x; Link next; }; Элемент (узел) двунаправленного списка Элемент списка записывается в виде структуры struct DListInt{ int x; DListInt *last; DListInt *next; };

практическое занятие (6 часа(ов)):

Практические примеры. Создание списка из чисел, введенных с клавиатуры // list-01.cpp : Создание линейного списка // #include "stdafx.h" #include <stdio.h> #include <iostream> using namespace std; // typedef struct ListInt *Link; struct ListInt{ int x; ListInt *next;// или Link next; }; int _tmain(int argc, _TCHAR* argv[]) { ListInt *head=NULL;// или Link head; ListInt *p; // или Link p; ListInt *q; // или Link q; int n; int a; // Данные с клавиатуры: cout<<"\n Size= "; cin>>n; if(n==0) {cout<<"\ List empty! \n"; return 1;} head = new ListInt; // Для головного элемента p = head; for (int i=0;i<n;i++) { cout<<"\n x="; cin>>a; p->x = a; q = p; p = new ListInt; q->next = p; } q->next = NULL; delete p; // Вывод списка: cout<<"\n List: \n"; p=head; while (p!=NULL) { cout<<"\n"<<p->x; p=p->next; } return 0; } Создание списка из чисел, введенных с клавиатуры // list-02.cpp : Создание двунаправленного списка // #include "stdafx.h" #include <stdio.h> #include <iostream> using namespace std; struct DListInt{ int x; DListInt *last; DListInt *next; }; int _tmain(int argc, _TCHAR* argv[]) { DListInt *head=NULL; DListInt *tail=NULL; DListInt *p; DListInt *q; int n; int a; // Данные с клавиатуры: cout<<"\n Size= "; cin>>n; if(n==0) {cout<<"\ List empty! \n"; return 1;} head = new DListInt; // Для головного элемента head->last = NULL; p = head; for (int i=0;i<n;i++) { cout<<"\n x="; cin>>a; p->x = a; q = p; p = new DListInt; // Назначаем связи: p->last = q; q->next = p; } q->next = NULL; tail = q; // удаляем элемент, созданный на последнем шаге: delete p; // Вывод списка с начала: cout<<"\n List: \n"; p=head; while (p!=NULL) { cout<<"\n"<<p->x; p=p->next; } // Вывод списка с конца: cout<<"\n List : \n"; p=tail; while (p!=NULL) { cout<<"\n"<<p->x; p=p->last; } return 0; }

Тема 45. Программирование в среде Visual Studio. Создание приложений с графическим интерфейсом

лекционное занятие (4 часа(ов)):

Описание среды программирования.

практическое занятие (4 часа(ов)):

Практические примеры.

4.3 Структура и содержание самостоятельной работы дисциплины (модуля)

N	Раздел Дисциплины	Семестр	Неделя семестра	Виды самостоятельной работы студентов	Трудоемкость (в часах)	Формы контроля самостоятельной работы
1.	Тема 1. История развития вычислительной техники, ее современное состояние	1		подготовка к дискуссии	2	дискуссия
2.	Тема 2. ЗАКОНЫ РАЗВИТИЯ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ	1		подготовка к дискуссии	2	дискуссия

N	Раздел Дисциплины	Семестр	Неделя семестра	Виды самостоятельной работы студентов	Трудоемкость (в часах)	Формы контроля самостоятельной работы
3.	Тема 3. КОМПЬЮТЕРНЫЕ СЕТИ. Адресация.	1		подготовка к контрольной работе	4	контрольная работа
4.	Тема 4. Интернет-технологии. Общие понятия.	1		подготовка к устному опросу	4	устный опрос
5.	Тема 5. Современный графический интерфейс пользователя (ГИП). Понятие графического компонента. Окно, типы окон, элементы окна, взаимное расположение окон, их перемещение, изменение размеров, пиктограммы. Рабочий стол. Запуск, переключение и завершение программ. Настройка рабочего стола. Работа с папками. Настройка параметров ГИП, экрана, мыши и клавиатуры. Шрифты, их установка. Работа с документами при помощи мыши и клавиатуры, обмен данными через буфер	1		подготовка к устному опросу	4	устный опрос
6.	Тема 6. Текстовые процессоры, основные приемы работы с ними. Шрифты, гарнитуры, их элементы и типы. Набор формул. Работа с таблицами и формами. Вставка и создание графики. Понятие стиля и шаблона. Драйверы принтеров, их установка и настройка, управление процессом печати.	1		подготовка домашнего задания	4	домашнее задание

N	Раздел Дисциплины	Семестр	Неделя семестра	Виды самостоятельной работы студентов	Трудоемкость (в часах)	Формы контроля самостоятельной работы
7.	Тема 7. Электронные таблицы, основные понятия. Рабочая книга, рабочий лист, ячейка, работа с ними. Создание формул, массивов и функций. Представление данных графиками и диаграммами. Анализ данных, итоги и сводные таблицы.	1		подготовка домашнего задания	4	домашнее задание
8.	Тема 8. Математические пакеты, их основные возможности. Вычисление выражений и функций, символьные преобразования, построение графиков. Встроенный язык программирования. Расширение ядра с помощью встраиваемых модулей и пакетов.	1		подготовка домашнего задания	8	домашнее задание
9.	Тема 9. Системы счисления: позиционные и непозиционные. Примеры систем счисления.	1		подготовка к контрольной работе	2	контрольная работа
10.	Тема 10. Алгоритм. Основные особенности алгоритма.	1		подготовка к устному опросу	2	устный опрос
11.	Тема 11. Запись алгоритмов. Блок-схемы.	1		подготовка домашнего задания	2	домашнее задание
12.	Тема 12. Рекуррентные соотношения. Вычисление рекуррентных соотношений.	1		подготовка к контрольной работе	4	контрольная работа
13.	Тема 13. Алгоритм вычисление сумм и произведений.	1		подготовка домашнего задания	4	домашнее задание

N	Раздел Дисциплины	Семестр	Неделя семестра	Виды самостоятельной работы студентов	Трудоемкость (в часах)	Формы контроля самостоятельной работы
14.	Тема 14. Алгоритмы вычисления минимальных и максимальных значений. Алгоритм вычисления количества элементов последовательности с заданным свойством.	1		подготовка домашнего задания	4	домашнее задание
15.	Тема 15. Синтаксис языка С. Операции арифметические, сравнения и логические, приоритет выполнения операций. Операции инкремента (++) и декремента (--). Оператор запятая. Оператор "?". Логические операции и оператор if.Оператор switch.Операторы цикла for(), while.	2		подготовка к устному опросу	6	устный опрос
16.	Тема 16. Массивы. Многомерные массивы	2		подготовка домашнего задания	4	домашнее задание
17.	Тема 17. Динамические массивы. Создание массивов с помощью malloc(). Создание массивов с помощью new.	2		подготовка домашнего задания	4	домашнее задание
18.	Тема 18. Арифметика указателей. Указатели и массивы.	2		подготовка домашнего задания	4	домашнее задание

N	Раздел Дисциплины	Семестр	Неделя семестра	Виды самостоятельной работы студентов	Трудоемкость (в часах)	Формы контроля самостоятельной работы
19.	Тема 19. Функции - правила создания. Объявление и определение функции. Формальные и фактические параметры. Передача параметров функции по значению и по ссылке. Массивы в качестве параметров функции. Аргументы по умолчанию. Рекурсия. Правила перегрузки функций. Указатель на функцию. Имя функции как параметр функции.	2		подготовка домашнего задания	8	домашнее задание
20.	Тема 20. Файлы - общие правила. Запись в текстовый файл. Чтение из текстового файла.	2		подготовка домашнего задания	8	домашнее задание
21.	Тема 21. Структуры.	2		подготовка к устному опросу	4	устный опрос
22.	Тема 22. Алгоритмы приближенного решения уравнений.	2		подготовка домашнего задания	4	домашнее задание
23.	Тема 23. Алгоритмы работы с матрицами.	2		подготовка домашнего задания	8	домашнее задание
24.	Тема 24. Алгоритмы решения систем линейных уравнений. Реализация метода исключения Гаусса. Вычисление определителей. Вычисление обратной матрицы.	2		подготовка домашнего задания	10	домашнее задание
25.	Тема 25. Повышение точности вычислений.	2		подготовка к дискуссии	2	дискуссия
27.	Тема 27. Классы. Атрибуты и операции. Реализация операций (функций-членов) класса вне класса. Инкапсуляция. Модификаторы доступа private и public.	3		подготовка к дискуссии	2	дискуссия

N	Раздел Дисциплины	Семестр	Неделя семестра	Виды самостоятельной работы студентов	Трудоемкость (в часах)	Формы контроля самостоятельной работы
28.	Тема 28. Конструкторы и деструкторы. Конструктор по умолчанию. Конструктор копирования.	3		подготовка домашнего задания	2	домашнее задание
29.	Тема 29. Объекты. Массив объектов. Инициализация массива объектов. динамическое выделение памяти объектам. Указатели на объекты. Указатель this. Передача объектов функции по ссылке и по значению.Дружественные функции. Возвращение объекта из функции.	3		подготовка домашнего задания	2	домашнее задание
30.	Тема 30. . Наследование. Правила наследования с модификаторами private и public. Модификатор доступа protected. Конструкторы и наследование. Вызов методов Множественное наследование.базового класса в производном.	3		подготовка домашнего задания	4	домашнее задание
31.	Тема 31. Виртуальные методы. Статическое и динамическое связывание.	3		подготовка к дискуссии	2	дискуссия
32.	Тема 32. Абстрактные классы.	3		подготовка к дискуссии	2	дискуссия
33.	Тема 33. Константные переменные. Указатель на константу. Константные параметры функций.	3		подготовка к дискуссии	2	дискуссия
35.	Тема 35. Перегрузка операций.	3		подготовка домашнего задания	2	домашнее задание
37.	Тема 37. Пространства имен.	3		подготовка к дискуссии	2	дискуссия

N	Раздел Дисциплины	Семестр	Неделя семестра	Виды самостоятельной работы студентов	Трудоемкость (в часах)	Формы контроля самостоятельной работы
41.	Тема 41. Алгоритмы внешней сортировки.	4		подготовка к дискуссии	2	дискуссия
	Итого				134	

5. Образовательные технологии, включая интерактивные формы обучения

Использование мультимедийной техники в процессе лекции и практических занятиях.

6. Оценочные средства для текущего контроля успеваемости, промежуточной аттестации по итогам освоения дисциплины и учебно-методическое обеспечение самостоятельной работы студентов

Тема 1. История развития вычислительной техники, ее современное состояние

дискуссия , примерные вопросы:

Обзор история развития вычислительной техники

Тема 2. ЗАКОНЫ РАЗВИТИЯ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

дискуссия , примерные вопросы:

Обсуждение законов развития

Тема 3. КОМПЬЮТЕРНЫЕ СЕТИ. Адресация.

контрольная работа , примерные вопросы:

Правила составления адресов

Тема 4. Интернет-технологии. Общие понятия.

устный опрос , примерные вопросы:

Основные конструкции HTML

Тема 5. Современный графический интерфейс пользователя (ГИП). Понятие графического компонента. Окно, типы окон, элементы окна, взаимное расположение окон, их перемещение, изменение размеров, пиктограммы. Рабочий стол. Запуск, переключение и завершение программ. Настройка рабочего стола. Работа с папками. Настройка параметров ГИП, экрана, мыши и клавиатуры. Шрифты, их установка. Работа с документами при помощи мыши и клавиатуры, обмен данными через буфер

устный опрос , примерные вопросы:

Демонстрация основных приемов работы

Тема 6. Текстовые процессоры, основные приемы работы с ними. Шрифты, гарнитуры, их элементы и типы. Набор формул. Работа с таблицами и формами. Вставка и создание графики. Понятие стиля и шаблона. Драйверы принтеров, их установка и настройка, управление процессом печати.

домашнее задание , примерные вопросы:

Выполнение заданий, включающих подготовку текстов с графикой, таблицами и формулами.

Тема 7. Электронные таблицы, основные понятия. Рабочая книга, рабочий лист, ячейка, работа с ними. Создание формул, массивов и функций. Представление данных графиками и диаграммами. Анализ данных, итоги и сводные таблицы.

домашнее задание , примерные вопросы:

Выполнение заданий с помощью среды MS Excel.

Тема 8. Математические пакеты, их основные возможности. Вычисление выражений и функций, символьные преобразования, построение графиков. Встроенный язык программирования. Расширение ядра с помощью встраиваемых модулей и пакетов.

домашнее задание , примерные вопросы:

Выполнение заданий по вычислениям

Тема 9. Системы счисления: позиционные и непозиционные. Примеры систем счисления.

контрольная работа , примерные вопросы:
Перевод чисел из различных систем счисления

Тема 10. Алгоритм. Основные особенности алгоритма.

устный опрос , примерные вопросы:
Основные особенности алгоритма

Тема 11. Запись алгоритмов. Блок-схемы.

домашнее задание , примерные вопросы:
Составление блок-схем

Тема 12. Рекуррентные соотношения. Вычисление рекуррентных соотношений.

контрольная работа , примерные вопросы:
Составление рекуррентных соотношений и реализация алгоритмов в виде блок-схем

Тема 13. Алгоритм вычисление сумм и произведений.

домашнее задание , примерные вопросы:
Реализация алгоритмов вычисления сумм и произведений

Тема 14. Алгоритмы вычисления минимальных и максимальных значений. Алгоритм вычисления количества элементов последовательности с заданным свойством.

домашнее задание , примерные вопросы:
Реализация алгоритмов вычисления минимальных и максимальных значений.

Тема 15. Синтаксис языка С. Операции арифметические, сравнения и логические, приоритет выполнения операций. Операции инкремента (++) и декремента (--). Оператор запятая. Оператор "?". Логические операции и оператор if.Оператор switch.Операторы цикла for(), while.

устный опрос , примерные вопросы:
Основные конструкции языка программирования С.

Тема 16. Массивы. Многомерные массивы

домашнее задание , примерные вопросы:
Задачи на массивы.

Тема 17. Динамические массивы. Создание массивов с помощью malloc(). Создание массивов с помощью new.

домашнее задание , примерные вопросы:
Задачи на динамические массивы.

Тема 18. Арифметика указателей. Указатели и массивы.

домашнее задание , примерные вопросы:
Задачи на использование указателей.

Тема 19. Функции - правила создания. Объявление и определение функции. Формальные и фактические параметры. Передача параметров функции по значению и по ссылке. Массивы в качестве параметров функции. Аргументы по умолчанию. Рекурсия. Правила перегрузки функций. Указатель на функцию. Имя функции как параметр функции.

домашнее задание , примерные вопросы:
Задачи на использование функций.

Тема 20. Файлы - общие правила. Запись в текстовый файл. Чтение из текстового файла.

домашнее задание , примерные вопросы:
Задачи с файлами.

Тема 21. Структуры.

устный опрос , примерные вопросы:

Основные правила использования структур.

Тема 22. Алгоритмы приближенного решения уравнений.

домашнее задание , примерные вопросы:

Реализация алгоритмов простой итерации, Ньютона, половинного деления.

Тема 23. Алгоритмы работы с матрицами.

домашнее задание , примерные вопросы:

Реализация алгоритмов сложения, умножения, транспонирования матриц и др.

Тема 24. Алгоритмы решения систем линейных уравнений. Реализация метода исключения Гаусса. Вычисление определителей. Вычисление обратной матрицы.

домашнее задание , примерные вопросы:

Реализация алгоритма исключения Гаусса.

Тема 25. Повышение точности вычислений.

дискуссия , примерные вопросы:

Приемы, позволяющие повысить точность вычислений.

Тема 26. Основные понятия объектно-ориентированного программирования (ООП).

Тема 27. Классы. Атрибуты и операции. Реализация операций (функций-членов) класса вне класса. Инкапсуляция. Модификаторы доступа private и public.

дискуссия , примерные вопросы:

Основные принципы ООП.

Тема 28. Конструкторы и деструкторы. Конструктор по умолчанию. Конструктор копирования.

домашнее задание , примерные вопросы:

Задачи на создание классов.

Тема 29. Объекты. Массив объектов. Инициализация массива объектов. динамическое выделение памяти объектам. Указатели на объекты. Указатель this. Передача объектов функции по ссылке и по значению.Дружественные функции. Возвращение объекта из функции.

домашнее задание , примерные вопросы:

Задачи на создание классов и использовании объектов.

Тема 30. . Наследование. Правила наследования с модификаторами private и public. Модификатор доступа protected. Конструкторы и наследование. Вызов методов Множественное наследование.базового класса в производном.

домашнее задание , примерные вопросы:

Правила наследования. Задачи на создание производных классов.

Тема 31. Виртуальные методы. Статическое и динамическое связывание.

дискуссия , примерные вопросы:

Особенности динамического связывания.

Тема 32. Абстрактные классы.

дискуссия , примерные вопросы:

Иерархия объектов - как создавать классы.

Тема 33. Константные переменные. Указатель на константу. Константные параметры функций.

дискуссия , примерные вопросы:

Когда и как использовать const.

Тема 34. Операции с файлами в C++ (классы ofstream и ifstream). Чтение из файла, запись в файл.

Тема 35. Перегрузка операций.

домашнее задание , примерные вопросы:

Задачи на перегрузку операций.

Тема 36. Раздельная компиляция, "стражи включения".

Тема 37. Пространства имен.

дискуссия , примерные вопросы:

Особенности использования пространств имен.

Тема 38. Структуры данных. Внутренняя и внешняя сортировки

Тема 39. Сортировка числового массива методом пузырька. Сортировка массива объектов методом пузырька. Сортировка числового массива методом просеивания. Сортировка числового массива методом Шелла. Сортировка числового массива методом быстрой сортировки.

Тема 40. Метод естественного слияния.

Тема 41. Алгоритмы внешней сортировки.

дискуссия , примерные вопросы:

Особенности внешней сортировки.

Тема 42. Алгоритмы поиска

Тема 43. Деревья цифрового поиска

Тема 44. Создание однонаправленного списка. Создание двунаправленного списка. Добавление нового узла в начало однонаправленного списка. Добавление нового узла в начало двунаправленного списка. Добавление нового узла в конец однонаправленного списка. Добавление нового узла в конец двунаправленного списка. Добавление нового узла в определенное место однонаправленного списка. Добавление нового узла в определенное место двунаправленного списка. Удаление первого узла однонаправленного списка. Удаление первого узла двунаправленного списка. Удаление последнего узла однонаправленного списка. Удаление последнего узла двунаправленного списка. Удаление заданного узла однонаправленного списка. Удаление заданного узла двунаправленного списка.

Тема 45. Программирование в среде Visual Studio. Создание приложений с графическим интерфейсом

Тема . Итоговая форма контроля

Тема . Итоговая форма контроля

Тема . Итоговая форма контроля

Тема . Итоговая форма контроля

Примерные вопросы к зачету и экзамену:

Вопросы к экзамену 2013

Синтаксис языка С

1. Структура программы (функция main(), директивы препроцессора, комментарии, операторы). Идентификаторы.
2. Типы данных. оператор sizeof().
3. Операции арифметические, сравнения и логические, приоритет выполнения операций.
4. Операция присваивания, составные операции присваивания.
5. Операции инкремента (++) и декремента (--).
6. Оператор запятая.
7. Оператор "?".
8. Логические операции и оператор if.
9. Оператор switch.
10. Операторы цикла for(), while.
11. Массивы. Многомерные массивы.
12. Динамические массивы. Создание массивов с помощью malloc(). Создание массивов с помощью new.

13. Указатели. Функция malloc(). Операторы new и delete. Инициализация динамической переменнной.
14. Арифметика указателей.
14. Указатели и массивы.
15. Функции - правила создания. Объявление и определение функции. Формальные и фактические параметры.
16. Передача параметров функции по значению и по ссылке.
17. Массивы в качестве параметров функции.
18. Аргументы по умолчанию.
19. Рекурсия.
20. Правила перегрузки функций.
21. Указатель на функцию. Имя функции как параметр функции.
22. Файлы - общие правила. Запись в текстовый файл. Чтение из текстового файла.

Конструкции языка C++

23. Структуры.
24. Классы. Атрибуты и операции. Реализация операций (функций-членов) класса вне класса.
25. Инкапсуляция. Модификаторы доступа private и public.
26. Конструкторы и деструкторы. Конструктор по умолчанию.
27. Объекты. Массив объектов. Инициализация массива объектов.
28. динамическое выделение памяти объектам. Указатели на объекты. Указатель this.
29. Передача объектов функции по ссылке и по значению. Возвращение объекта из функции.
30. Дружественные функции.
31. Конструктор копирования.
32. Конструктор преобразования.
33. Статические элементы класса.
34. Наследование. Правила наследования с модификаторами private и public.
35. Модификатор доступа protected.
36. Конструкторы и наследование.
37. Вызов методов базового класса в производном.
38. Множественное наследование.
39. Виртуальные методы. Статическое и динамическое связывание.
40. Абстрактные классы.
41. Константные переменные. Указатель на константу. Константные параметры функций.
42. Операции с файлами в C++ (классы ofstream и ifstream). Чтение из файла, запись в файл.
43. Перегрузка операций.
44. Раздельная компиляция, "стражи включения".
45. Пространства имен.

Структуры данных

1. Внутренняя и внешняя сортировки
2. Сортировка числового массива методом пузырька.
3. Сортировка массива объектов методом пузырька.
4. Сортировка числового массива методом просеивания.
5. Сортировка числового массива методом Шелла.
6. Сортировка числового массива методом быстрой сортировки.
7. Метод естественного слияния.
8. Создание однонаправленного списка.
9. Создание двунаправленного списка.

10. Создание однонаправленного списка из чисел, прочитанных из файла.
11. Создание двунаправленного списка из чисел, прочитанных из файла.
12. Добавление нового узла в начало однонаправленного списка.
13. Добавление нового узла в начало двунаправленного списка.
14. Добавление нового узла в конец однонаправленного списка.
15. Добавление нового узла в конец двунаправленного списка.
16. Добавление нового узла в определенное место однонаправленного списка.
17. Добавление нового узла в определенное место двунаправленного списка.
18. Удаление первого узла однонаправленного списка.
19. Удаление первого узла двунаправленного списка.
20. Удаление последнего узла однонаправленного списка.
21. Удаление последнего узла двунаправленного списка.
22. Удаление заданного узла однонаправленного списка.
23. Удаление заданного узла двунаправленного списка.

7.1. Основная литература:

Технология программирования. Базовые конструкции C/C++, Липачёв, Евгений Константинович, 2012г.

1.Литвиненко Н. А. Технология программирования на C++. Win32 API-приложения. ? СПб.: БХВ-Петербург, 2010. ? 280 с.: http://znanium.com/bookread.php?book=351463

2.Машнин Т. С. Современные Java-технологии на практике. ? СПб.: БХВ-Петербург, 2010. ? 560 с. URL: http://znanium.com/bookread.php?book=351236

3.Хабибуллин И. Ш. Самоучитель Java / Ильдар Хабибуллин. ? 3-е изд., перераб. и доп. ? СПб.: БХВ-Петербург, 2008. ? 758 с.: http://znanium.com/bookread.php?book=350488

7.2. Дополнительная литература:

Технология Java, Хабибуллин, Ильдар Шаукатович, 2010г.

Информационные системы, Избачков, Юрий Сергеевич;Петров, В.Н., 2006г.

1.Немцова Т. И. Программирование на языке высокого уровня. Программир. на языке C++: Уч. пос. / Т.И.Немцова и др.; Под ред. Л.Г.Гагариной - М.: ИД ФОРУМ: ИНФРА-М, 2012. - 512 с.: http://znanium.com/bookread.php?book=244875

7.3. Интернет-ресурсы:

Комбинаторные алгоритмы для программистов - http://www.intuit.ru/studies/courses/65/65/info

Основы объектно-ориентированного программирования - http://www.intuit.ru/studies/courses/71/71/info

Основы программирования на языке C - http://www.intuit.ru/studies/courses/43/43/info

Современные офисные приложения - http://www.intuit.ru/studies/courses/81/81/info

Язык программирования C++ - http://www.intuit.ru/studies/courses/17/17/info

C++ Lessons and Topics - http://functionx.com/cpp/index.htm

CyberForum.ru - форум программистов и сисадминов - http://www.cyberforum.ru/

HTML Lessons - http://functionx.com/html/index.htm

Visual C++ - http://msdn.microsoft.com/ru-ru/library/60k1461a.aspx

Клуб программистов - http://programmersforum.ru/

8. Материально-техническое обеспечение дисциплины(модуля)

Освоение дисциплины "Технологии программирования и работа на электронных вычислительных машинах" предполагает использование следующего материально-технического обеспечения:

Мультимедийная аудитория, вместимостью более 60 человек. Мультимедийная аудитория состоит из интегрированных инженерных систем с единой системой управления, оснащенная современными средствами воспроизведения и визуализации любой видео и аудио информации, получения и передачи электронных документов. Типовая комплектация мультимедийной аудитории состоит из: мультимедийного проектора, автоматизированного проекционного экрана, акустической системы, а также интерактивной трибуны преподавателя, включающей тач-скрин монитор с диагональю не менее 22 дюймов, персональный компьютер (с техническими характеристиками не ниже Intel Core i3-2100, DDR3 4096Mb, 500Gb), конференц-микрофон, беспроводной микрофон, блок управления оборудованием, интерфейсы подключения: USB,audio, HDMI. Интерактивная трибуна преподавателя является ключевым элементом управления, объединяющим все устройства в единую систему, и служит полноценным рабочим местом преподавателя. Преподаватель имеет возможность легко управлять всей системой, не отходя от трибуны, что позволяет проводить лекции, практические занятия, презентации, вебинары, конференции и другие виды аудиторной нагрузки обучающихся в удобной и доступной для них форме с применением современных интерактивных средств обучения, в том числе с использованием в процессе обучения всех корпоративных ресурсов. Мультимедийная аудитория также оснащена широкополосным доступом в сеть интернет. Компьютерное оборудованием имеет соответствующее лицензионное программное обеспечение.

Компьютерный класс, представляющий собой рабочее место преподавателя и не менее 15 рабочих мест студентов, включающих компьютерный стол, стул, персональный компьютер, лицензионное программное обеспечение. Каждый компьютер имеет широкополосный доступ в сеть Интернет. Все компьютеры подключены к корпоративной компьютерной сети КФУ и находятся в едином домене.

Учебно-методическая литература для данной дисциплины имеется в наличии в электронно-библиотечной системе Издательства "Лань" , доступ к которой предоставлен студентам. ЭБС Издательства "Лань" включает в себя электронные версии книг издательства "Лань" и других ведущих издательств учебной литературы, а также электронные версии периодических изданий по естественным, техническим и гуманитарным наукам. ЭБС Издательства "Лань" обеспечивает доступ к научной, учебной литературе и научным периодическим изданиям по максимальному количеству профильных направлений с соблюдением всех авторских и смежных прав.

Электронное издание учебно-справочного пособия лектора.

Программа составлена в соответствии с требованиями ФГОС ВПО и учебным планом по направлению 010800.62 "Механика и математическое моделирование" и профилю подготовки Общий профиль .

Автор(ы):
Липачев Е.К. _____
"__" _____ 201 __ г.

Рецензент(ы):
Маклецов С.В. _____
"__" _____ 201 __ г.