Министерство образования и науки РФ
Рубцовский индустриальный институт (филиал)
ФГБОУ ВПО «Алтайский государственный технический университет им. И.И. Ползунова»

Н.А.ЛАРИНА

ПРОГРАММИРОВАНИЕ

Часть II
(язык программирования Си в задачах)

Методическое пособие для студентов направления 230100 «Информатика и вычислительная техника» дневной формы обучения

Рубцовск 2013

УДК 004.438

Л 25

Ларина Н.А. Программирование. Часть II (язык программирования Си в задачах): Методическое пособие для студентов направления 230100 «Информатика и вычислительная техника» дневной формы обучения /Н.А. Ларина. Рубцовский индустриальный институт. – Рубцовск, 2013. – 115 с.

В пособии содержится теоретический, практический и справочный материалы по предмету «Программирование». Кроме того, включена памятка студенту, графики практической и самостоятельной работ по данному предмету.

Пособие предназначено для обучения студентов, обладающих навыками пользовательской работы на персональном компьютере, основными понятиями и методами современного практического программирования.

Выражаю благодарность А.Г. Лебедеву и Ю.В. Сахань за предоставленные задания для курсовых и лабораторных работ.

Рассмотрено и одобрено
на заседании НМС.
Протокол № 9 от 19.12.13

Рецензент: к.ф.-м.н., доцент, зав. кафедрой ВМФиХ РИИ Г.А. Обухова

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Дисциплина «Программирование» (Си) предназначена для приобретения навыков решения задач различной сложности математического и экономического содержания, средствами языка СИ с применением ЭВМ. А также расширения и углубления знаний в области теории алгоритмов, программирования и данных, навыков программирования, приближая методы программирования к машинным командам.

Язык Си изучается после освоения программирования на языке Паскаль (Pascal) в первом семестре.

Целью изучения рассматриваемой дисциплины является теоретическая и практическая подготовка студентов в части процесса разработки программ, взаимосвязи между аппаратными и программными ресурсами, системными программными средствами и прикладными программами пользователя. Ознакомить с особенностями архитектуры и основами программирования в соответствии с ними.

Дисциплина «Программирование» (Си) входит в цикл общепрофессиональных дисциплин федерального компонента ГОС ВПО специальности 230100 «Информатика и вычислительная техника».

Предметом изучения курса является функциональное программирование на языке Си в среде современных операционных систем семейства Windows. Программа курса разбита на 4 модуля: (1) Введение в программирование на языке Си; (2) Основы программирования с применением массивов; (3) Организация ввода-вывода на языке Си и (4) Программирование динамических объектов и графики.

После изучения курса студент получает достаточно полное представление о содержании современного программирования, обустройстве современных операционных систем. На практических и лабораторных занятиях вырабатываются навыки программирования на Си в интегрированной среде разработки Borland C.

В результате изучения дисциплины студенты должны:

***Знать*:**

- основные конструкции языка Си;
- классификацию типов данных;
- операторов языка, их классификацию и приоритет выбора;
- определение, описание и применение стандартных и внутренних функций;
- основные динамические объекты программирования: списки одно и двух связный, стек, очередь, дерево, дек;
- графические функции и их применение;
- методику и основные приемы программирования, сборки программ и проектов в среде Borland Си.

Уметь:

- правильно определить необходимые и достаточные входные, выходные и промежуточные данные, их тип и видимость;
- соблюдать синтаксические и семантические нормы языка;
- разрабатывать программы на языке Си на основании заданного и самостоятельно составленного алгоритма;
- создавать элементарный проект;
- пользоваться функциями для вывода графических изображений (статических и динамических);
- разрабатывать задачи, с применением для их решения, приближенных методов вычисления;
- применять динамические объекты в решении задач.

Приобрести навыки:

- составления контрольных примеров;
- отладки программ;
- доказательства правильности решения задачи с использованием контрольных примеров;

- разработки простейших программ для реализации вычислительных методов математики средней сложности;
- документального описания решения задачи.

Общий объем дисциплины составляет 125 часов, которые распределены следующим образом:

Номер семестра	Аудиторные занятия				СРС	Наличие курсовых проектов (КП), курсовых работ (КР), расчетных заданий (РЗ)	Форма итоговой аттестации
	Всего	Лекции	Лаб.р.	Практ.			
2	102	34	34	34	57	КР	Экзамен

Виды и содержание занятий по дисциплине

ЛЕКЦИИ

МОДУЛЬ 1

Лекция 1. Введение в алгоритмический язык Си (2 ч [1,2]).

История языка, его особенности. Структура программы Си. Константы.

Лекция 2. Простые типы данных (2 ч [1,2,4]).

Целый, плавающий, символьный типы. Перечислимый тип (перечисления). Операции языка Си.

Лекция 3. Операторы Си (2ч [2,4,6]).

Условные операторы. Операторы цикла. Вспомогательные управляющие операторы.

МОДУЛЬ 2

Лекция 4. Массивы и указатели. Сложные структуры данных (2ч [2,4,6]).

Классы памяти. Массивы. Указатели. Структуры (записи). Объединения (смеси).

Лекция 5. Препроцессор и макросы (1ч [2])

Макроопределения и препроцессор. Условная компиляция. Прагмы.

МОДУЛЬ 3

Лекция 6. Организация ввода-вывода (4ч [1,2,4-6]).

Ввод-вывод трех уровней. Ввод-вывод нижнего уровня. Функции ввода-вывода верхнего уровня (потока). Управление буферизацией. Двоичный и текстовый режимы. Ввод-вывод для консоли и порта. Форматизованный ввод-вывод. Функции обработки строк.

МОДУЛЬ 4

Лекция 7. Функции в Си. Динамические объекты (2ч [4,6]).

Функции. Передача параметров. Организация проекта. Указатели. Управление динамической памятью. Динамически связанные списки.

Лекция 8. Графика (2ч [6]).

Инсталляция графического режима. Функции построения фигур. Перемещение фигур по экрану. Использование окон.

ПРАКТИЧЕСКИЕ ЗАНЯТИЯ

МОДУЛЬ 1

Практическое занятие №1. Структура программы Си. (2ч) Упражнения с использованием различных типов данных, операций ввода и вывода данных. Операции присваивания. Упражнения на вычисление значений функций.

Практическое занятие №2. Операции языка Си. (2ч) Унарные, бинарные, тернарная операции. Поразрядные операции.

Практическое занятие №3. Основные операторы (2ч) Сложные и составные операторы. Упражнения с использованием операторов сравнения, операторов цикла.

МОДУЛЬ 2

Практическое занятие №4. Одномерные массивы (2ч) Упражнения на описание и заполнение массивов.

Практическое занятие №5. Строки. Матрицы (2ч) Упражнения на описание и заполнение строковых массивов и массивов различной размерности.

Практическое занятие №6. Структуры, объединения (2ч) Упражнения на описание и заполнение массивов от записи, использование вложенных структур и объединений.

МОДУЛЬ 3

Практическое занятие №7. Работа с файлами (2ч) Составление программ на создание, чтение и перезапись файлов различных типов.

МОДУЛЬ 4

Практическое занятие №8. Функции, динамические структуры (3ч) Упражнения на описание динамических структур. Программирование работы со стеком. Упражнения на составление постфиксных записей алгебраических выражений. Создание списка в динамической памяти. Использование подпрограмм.

ЛАБОРАТОРНЫЕ РАБОТЫ

МОДУЛЬ 1

Лабораторная работа 1. Структура программы Си (2ч)

Лабораторная работа 2. Основные операции (2ч)

Лабораторная работа 3. Операторы Си (2ч)

Лабораторная работа 4. Операторы Си (2ч)

МОДУЛЬ 2

Лабораторная работа 5. Работа с массивами (2ч)

Лабораторная работа 6. Работа с массивами (2ч)

Лабораторная работа 7. Работа со структурами в Си (2ч)

Лабораторная работа 8. Работа с объединением в Си (2ч)

Лабораторная работа 9. Препроцессор и организация макросов в Си (2ч)

МОДУЛЬ 3

Лабораторная работа 10. Текстовые файлы (2ч)

Лабораторная работа 11. Прямой доступ (2ч)

МОДУЛЬ 4

Лабораторная работа 12. Функции в Си (2ч)

Лабораторная работа 13. Работа с динамическими структурами (2ч)

Лабораторная работа 14. Организация проекта в Си (2ч)

Лабораторная работа 15. Организация проекта в Си (2ч)

Лабораторная работа 16. Графика (2ч)

Лабораторная работа 17. Графика (2ч)

КУРСОВАЯ РАБОТА

Курсовая работа предполагает составление и отладку программы, реализующей один из приближенных методов вычислительной математики.

В курсовой работе должны быть использованы:

- основные приемы составления программ Си;
- организация функций;
- методы организации окон и сервиса пользователя.

В отчете по курсовой работе должны быть следующие разделы: постановка задачи, алгоритм решения, руководство пользователя, руководство системного программиста, листинг программы, контрольный пример и доказательство правильности решения задачи (Приложение Ж).

САМОСТОЯТЕЛЬНАЯ РАБОТА СТУДЕНТОВ

Самостоятельная работа студентов составляет 57ч. и заключается в:

№ п/п	Перечень самостоятельных занятий	Объем СРС час/вес	Рекомендуемая литература
1.	Подготовка к практическим занятиям	17/0,3	[1-11]
2.	Подготовка к лабораторным работам	17/0,3	[1,2,4-7]
3.	Изучение дополнительной литературы	10/0,18	[4-6,9-11]
4.	Выполнение курсовой работы	13/0,22	[8]
	ИТОГО:	57/1,00	

Формы и содержание текущей и итоговой оценки по дисциплине

Форма итоговой аттестации: 2 семестр – экзамен.

Содержание текущей и итоговой аттестации раскрывается в комплекте контролирующих материалов, предназначенных для проверки соответствия уровня подготовки по дисциплине требованиям ГОС ВПО и СТП.

В течение 2 семестра текущий контроль осуществляется по тестам текущего контроля знаний по дисциплине (приложение А).

Экзамен (2 семестр) принимается в письменной форме по билетам, примерные вопросы к которым даны в приложении Б.

Контроль остаточных знаний проводится по тестам (приложение В).

В течение 2 семестра с целью закрепления теоретических знаний студенты посещают практические занятия, задания для которых приведены в приложение Г. Наряду с практическими, проводятся лабораторные занятия, задания к которым содержатся в приложении Е.

К концу семестра студенты выполняют и защищают курсовую работу.

При изучении дисциплины используется рейтинговая система оценки учебной работы студента, соответствующая Положению «О модульно-рейтинговой системе квалиметрии учебной деятельности студентов» СМК ОПД 01-19-2008. Памятка (силлабус) дисциплины приведена в приложении Д.

Рейтинговая система оценки индивидуальной учебной деятельности студентов

При изучении дисциплины «Программирование» (Си) используется рейтинговая система оценки учебной работы студента.

В РИИ АлтГТУ принята 100-балльная система оценок. Данные оценки учитываются при подсчете рейтингов, назначении стипендии и в других случаях. Традиционная шкала отметок используется в только в зачетных книжках. Соответствие оценок устанавливается следующим образом:

Баллы	Оценка
75-100	Отлично
50-74	Хорошо
25-49	Удовлетворительно
0-24	Неудовлетворительно

Успеваемость студента оценивается с помощью текущего рейтинга (во время аттестации и в конце семестра) и итогового рейтинга после сессии. Во всех случаях рейтинг вычисляется по формуле:

$$R_T = \frac{\sum R_i p_i}{\sum p_i},$$

где R_i - оценка за i-тую контрольную точку, p_i - вес этой контрольной точки. Суммирование проводится по всем контрольным точкам с начала семестра до момента вычисления рейтинга.

Оценка индивидуальной деятельности студентов по дисциплине производится согласно графику контроля, приведенному в памятке для студента (приложение Д). Оценка индивидуальной деятельности студентов по дисциплине складывается из следующих видов работ (макс. число баллов 100, в долях единицы - 1).

График контроля

Контрольное испытание	Время проведения	Вес в итоговом рейтинге (балл)	Примечания
Практические занятия	1-17 неделю	0,25 (25)	17 практических занятий
Лабораторные работы	1-17 неделю	0,25 (25)	17 лабораторных работ
Посещение занятий	1-17 неделю	0,1 (10)	Оценивается отсутствие пропусков занятий, активность студентов на лекции
Курсовая работа	В конце семестра	0,2 (20)	Предоставляется отчёт
Экзамен	Сессия	0,2 (30)	2 вопроса в билете – теоретический и практический

Для допуска к экзамену студент должен в течение семестра набрать не менее 25 баллов.

График аудиторных занятий и самостоятельной работы студентов

по специальности 230100 «Информатика и вычислительная техника»

на 2 семестр

Наименование вида работ	Недели семестра																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1 Аудиторные занятия:																	
- лекции	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
- лабораторные занятия	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
- практические занятия	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
2 Самостоятельная работа:																	
- подготовка к лекциям	постоянно																
- подготовка к лабораторным работам	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
- подготовка к текущей аттестации				+	+					+	+					+	+
3 Формы текущей аттестации:																	
- коллоквиум (КЛ)																	
- контрольная работа (К)																	
- контрольный опрос (КО)																	
- защита лабораторной работы (ЗР)																	
- другие виды аттестации (подготовка и защита курсовой работы)											+	+	+	+	+	+ Ззащита 0,2	
4 Посещение занятий	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	Конспект 0,1
5 Итоговая аттестация	Экзамен во время сессии; вес 0,2																

Специфические особенности дисциплины

Специфика преподавания данной дисциплины выражается следующими положениями: необходимо первоначальное изучение работы с приложениями Microsoft Office (Word, Excel, Power Point) для оформления отчёта о курсовой работе; реализация самостоятельной работы, в рамках курсовой, требует от студентов большого количества дополнительных знаний и умений, например, из области алгоритмизации, информационных технологий, дизайна, стандартов и др., которые студенты изучают самостоятельно.

Организация самостоятельной работы студентов

Самостоятельная работа студентов по дисциплине включает в себя подготовку к очередным лекциям, лабораторным и практическим работам, контрольным точкам, защите курсовой работы и экзамену. Ежедневная подготовка требует от студента глубокого изучения теоретического материала лекций.

На организацию СРС по выполнению самостоятельной работы отводятся определенные часы из расчёта общей нагрузки. Теоретическая база подкрепляется самостоятельным освоением рекомендованной кафедрой литературой в читальных залах или библиотеке РИИ, работой с Internet- ресурсами.

Для подготовки к практическим - лабораторным занятиям используются учебные пособия (см. перечень учебно-методической литературы).

Элементы научного поиска при изучении дисциплины

При изучении дисциплины используются следующие формы привлечения студентов к самостоятельной творческой деятельности:

- на лабораторных, практических и самостоятельных работах студентам предлагаются творческие задания по разработке алгоритма и решения разнообразных по тематике задач;
- элементом научного поиска является сбор, обработка и способы хранения материалов по изучаемому предмету, сбор функций для решения классических задач программирования.

ИСТОРИЯ ЯЗЫКА СИ

Сейчас, наверно, невозможно найти в мире специалиста в IT-области, который бы не слышал о языке Си. Этот язык приобрёл огромную популярность во всём мире и оказал значительное влияние на многие другие языки программирования. Именно он является предшественником таких языков, как C++, C#, Java; менее известных (например, J#). Компания Microsoft для разработки родного языка к своей платформе .Net выбрала именно Си-подобный синтаксис. Язык Си изменил жизнь программистов прошлого века и применялся в области низкоуровневого программирования, оставив ассемблеру только те места, где производительность имела критическое значение.

Имена создателей языка Си — Кен Томпсон и Денис Ритч, но язык Си уходит корнями к языку ALGOL (расшифровывается как ALGorithmic Language), который был разработан в 1958 году совместно с Комитетом европейских и американских учёных в сфере компьютерных наук на встрече в 1958 в Швейцарской высшей технической школе Цюриха. Язык был ответом на некоторые недостатки языка FORTRAN и попыткой их исправить.

На базе языка ALGOL-60 математическая лаборатория Кембриджского университета совместно с Компьютерным отделом Лондонского университета создали в 1963 году язык CPL (Combined Programming Language).

Язык CPL посчитали сложным, и Мартином Ричардсоном был создан в 1966 году язык BCPL, основное предназначение которого заключалось в написании компиляторов. Сейчас он практически не используется, но в своё время он играл важную роль.

BCPL послужил предком для языка Би, разработанного в 1969 в AT&T Bell Telephone Laboratories Кеном Томпсоном и Денсом Ритчи.

На языке Би были написаны самые ранние версии UNIX, созданной как ответ на проект Multics, разрабатываемый в той же Bell Laboratories. Именно этот язык послужил непосредственным предшественником языка Си.

По поводу возникновения языка Си Питер Мойлан в своей книге «The case against C» пишет: «Нужен был язык, способный обойти некоторые жесткие

правила, встроенные в большинство языков высокого уровня и обеспечивающие их надежность. Нужен был такой язык, который позволил бы делать то, что до него можно было реализовать только на ассемблере или на уровне машинного кода». Си стал именно таким языком. Это обусловило его дальнейшую популярность в таких отраслях программирования, как написание драйверов и прочих аспектах низкоуровневого программирования.

Язык программирования Си был разработан в стенах Bell Labs в период с 1969 по 1973 годы. Как признался сам Ритчи, самый активный период творчества приходился на 1972 год.

За всё время своего существования язык Си оброс легендами по поводу мотивов своего создания. Согласно одной из легенд, Керниган и Ритчи любили одну компьютерную игру, которую они запускали на главном сервере компании. Позже они захотели перенести её на компьютер, стоящий в офисе. Но он, к сожалению, не имел операционной системы, что сподвигло Кернигана и Ритчи её написать. Когда они захотели перенести систему на другой компьютер, это оказалось непростой задачей, так как система была написана полностью на ассемблере. Тогда у них возникла идея переписать её на язык высокого уровня. Сначала для этих целей планировали использовать язык Би, но в связи с тем, что он не давал на полную использовать новые возможности компьютера, на который они хотели перенести систему, было решено создать свой язык.

Согласно другой легенде, язык Си был первоапрельской шуткой, которая обрела нешуточную популярность.

Компилятор языка Си унаследовал традицию, заложенную ещё Никлаусом Виртом, и был написан на самом Си. Согласно мнению большинства, название языка Си является третьей буквой алфавита, как указание на то, что язык Си является более усовершенствованным, чем язык Би. Сам Ритчи по поводу названия языка говорил следующее: «Создав систему типов, соответствующий синтаксис и компилятор для нового языка, я почувствовал, что он заслуживает нового имени: NB показалось мне недостаточно четким. Я решил следовать однобуквенному

стилю и назвал его C (Си), оставляя открытым вопрос, являлось ли после B это следующей буквой в алфавите или в названии BCPL».

Успех Си в основном связан с тем, что на нём была написана значительная часть операционной системы UNIX, которая в итоге приобрела очень большую популярность. Если считать по количеству используемых на данный момент операционных систем, разработанных на базе UNIX, то она является самой распространённой системой в мире. В связи с её распространённостью, а также с тем, что на данный момент объём операционной системы измеряется в миллионах строк кода (для примера, в последних версиях Linux содержится более 10 000 000 строк кода), задача о переписывании UNIX на другой язык становится практически невыполнимой (также следует учитывать тот факт, что при ручном переписывании неизбежно возникнут ошибки, что существенно снизит стабильность работы, а при переводе с использованием программных средств пострадает производительность кода). Кроме того, язык Си, будучи приближённым к аппаратной реализации компьютера, позволяет выжать из него намного больше, чем многие другие языки программирования. Это обстоятельство показывает бессмысленность перевода UNIX на другой язык. Таким образом, если другие языки программирования могут исчезнуть с течением времени, уступив дорогу новым технологиям, то язык *Си будет жить,* пока живёт UNIX. То есть *пока существуют компьютеры* в том виде, в котором мы их себе представляем.

В то время как Си набирал всё большую популярность, компиляторы для него выпускались различными фирмами, и зачастую программа, которая компилировалась на компиляторе одной фирмы, не компилировалась на компиляторе другой. Всё это было связано с отсутствием чётко оговоренного стандарта языка Си. Все разработчики ориентировались на книгу Кернигана и Ритчи, но интерпретировали её по-своему.

Разработкой стандарта языка Си занялся Американский национальный институт стандартов (ANSI). При нём в 1983 году был сформирован комитет X3J11, который занялся разработкой стандарта. Первая версия стандарта была выпущена в 1989 году и получила название C89. В 1990, внеся небольшие

изменения в стандарт, его приняла Международная Организация Стандартизации ISO. Тогда он стал известен под кодом ISO/IEC 9899:1990, но в среде программистов закрепилось название, связанное с годом принятия стандарта: C90. Последней на данный момент версией стандарта является стандарт ISO/IEC 9899:1999, также известный как C99, который был принят в 2000 году.

Среди нововведений стандарта C99 следует обратить внимание на изменение правила, касающегося места объявления переменных. Теперь новые переменные можно было объявлять посреди кода, а не только в начале составного блока или в глобальной области видимости. Это уводит Си от концепции объявления переменных в начале функции, которая присутствует в Pascal, но даже с принятием стандарта C99, в программе Borland Embarcadeo RAD Studio 2010, ограничение на объявление переменных в начале блока кода всё ещё действует. Также можно указать другие места, в которых стандарты Си не до конца соблюдаются. Возможно, это связано с тем, что основное внимание больших компаний, таких как Microsoft и Borland, сосредоточено на более новых языках программирования. Однако, согласно заверениям компании Sun, Microsystems, её среда разработки Sun Studio полностью поддерживает C99.

Стандарт C99 сейчас в большей или меньшей степени поддерживается всеми современными компиляторами языка Си. В идеале код, написанный на Си с соблюдением стандартов и без использования аппаратно и системно-зависимых вызовов, становился как аппаратно, так и платформенно-независимым кодом.

В 2007 году начались работы над следующим стандартом языка Си: C1x.

ХАРАКТЕРИСТИКИ ЯЗЫКА

Си (англ. «*C*») - стандартизированный процедурный, компилируемый язык программирования. Для языка Си характерны лаконичность, стандартный набор конструкций управления потоком выполнения, структур данных и обширный набор операций.

Основные реализации языка: GCC, TCC, Turbo C, Watcom, Oracle Solaris Studio C. К диалектам языка стоит отнести: «K&R» C, ANSI C, C90, C99, C11.

Язык программирования Си отличается минимализмом. Авторы языка хотели, чтобы программы на нём легко компилировались с помощью однопроходного компилятора, чтобы каждой элементарной составляющей программы после компиляции соответствовало *весьма небольшое* число машинных команд. Однопроходный компилятор компилирует программу, не возвращаясь назад, к уже обработанному тексту. Поэтому использованию функции и переменных должно предшествовать их объявление. Код на Си можно легко писать на низком уровне абстракции, почти как на ассемблере. Иногда Си называют «универсальным ассемблером» или «ассемблером высокого уровня», что отражает различие языков ассемблера для разных платформ и единство стандарта Си, код которого может быть скомпилирован без изменений практически на любой модели компьютера. Си часто называют языком среднего уровня или даже низкого уровня, учитывая то, как близко он работает к реальным устройствам. Но, в строгой классификации, он является языком высокого уровня.

Компиляторы Си разрабатываются сравнительно легко благодаря простоте языка и малому размеру стандартной библиотеки. Поэтому данный язык доступен на самых различных платформах (возможно, круг этих платформ шире, чем у любого другого существующего языка). К тому же, несмотря на свою низкоуровневую природу, язык позволяет создавать переносимые программы и поддерживает в этом программиста. Программы, соответствующие стандарту языка, могут компилироваться на самых различных компьютерах.

Си создавался с одной важной целью: сделать более простым написание больших программ с минимумом ошибок по правилам процедурного

программирования, не добавляя на итоговый код программ лишних накладных расходов для компилятора, как это всегда делают языки очень высокого уровня, такие как Бейсик. С этой стороны, Си предлагает следующие важные особенности:

- простую языковую базу, из которой вынесены в библиотеки многие существенные возможности, например математические функции или функции управления файлами и даже функции ввода-вывода;
- ориентацию на процедурное программирование, обеспечивающую удобство применения структурного стиля программирования;
- систему типов, предохраняющую от бессмысленных операций;
- использование препроцессора для, например, определения макросов и включения файлов с исходным кодом;
- непосредственный доступ к памяти компьютера с использованием указателей;
- минимальное число ключевых слов;
- передачу параметров в функцию по значению, а не по ссылке (при этом передача по ссылке эмулируется с помощью указателей);
- указатели на функции и статические переменные;
- области действия имён;
- структуры и объединения - определяемые пользователем составные типы данных, которыми можно манипулировать как одним целым;

Одним из последствий высокой эффективности и переносимости Си стало то, что многие компиляторы, интерпретаторы и библиотеки других языков высокого уровня часто написаны на языке Си.

НАБОР ИСПОЛЬЗУЕМЫХ СИМВОЛОВ

Язык Си был создан уже после внедрения стандарта ASCII, поэтому использует почти все его графические символы (нет только `$ @ `). Более старые языки вынуждены были обходиться более скромным набором — так, Фортран, Лисп и Кобол использовали только круглые скобки `()`, а в Си есть и круглые `()`, и квадратные `[]`, и фигурные `{}`. Кроме того, в Си различаются заглавные и строчные буквы.

КОММЕНТАРИИ

Текст, заключённый в служебные символы /* и */ (в этом порядке), полностью игнорируется компилятором. Вложение комментариев не допускается. Компиляторы, совместимые со стандартом C99, также позволяют использовать комментарии, начинающиеся с символов // и заканчивающиеся переводом строки

КЛЮЧЕВЫЕ СЛОВА

В C89 есть 32 ключевых слова:

auto	Double	int	struct
break	Else	Long	switch
case	Enum	register	typedef
char	Extern	return	union
const	Float	short	unsigned
continue	For	signed	void
default	Goto	sizeof	volatile
do	If	static	while

C99 добавляет пять новых: _Bool, inline, _Complex, _Imaginary, restrict.

КОМПИЛЯЦИЯ ПРОГРАММ

Программа на языке Си - один или несколько текстовых файлов, которые также называются исходными.

Исполнить исходные файлы нельзя, их необходимо скомпилировать, т.е. создать исполняемый файл, содержащий в себе инструкции процессора и пригодный для запуска на компьютере.

Процесс преобразования исходных файлов в исполняемый называется компиляцией. В результате получится файл hello.exe. Этот файл является исполняемым, и его можно запускать на выполнение.

СТРУКТУРА ПРОГРАММЫ НА ЯЗЫКЕ СИ

Определение: Правила написанная текста на языке программирования, называются синтаксисом.

Программа на языке Си состоит из модулей, которые, в свою очередь, состоят из функций.

Функция представляет собой обычную программу. Термин «функция» в языке Си охватывает понятия: «подпрограмма», «процедура», и «функция», используемые в других языках программирования. Функции создаются в виде файлов, из которых затем образуется модуль.

Программа на языке Си может содержать одну функцию или, что бывает чаще, большее одной. Исполняемая программа *.exe* образуется из модулей с помощью специальной программы – загрузчика.

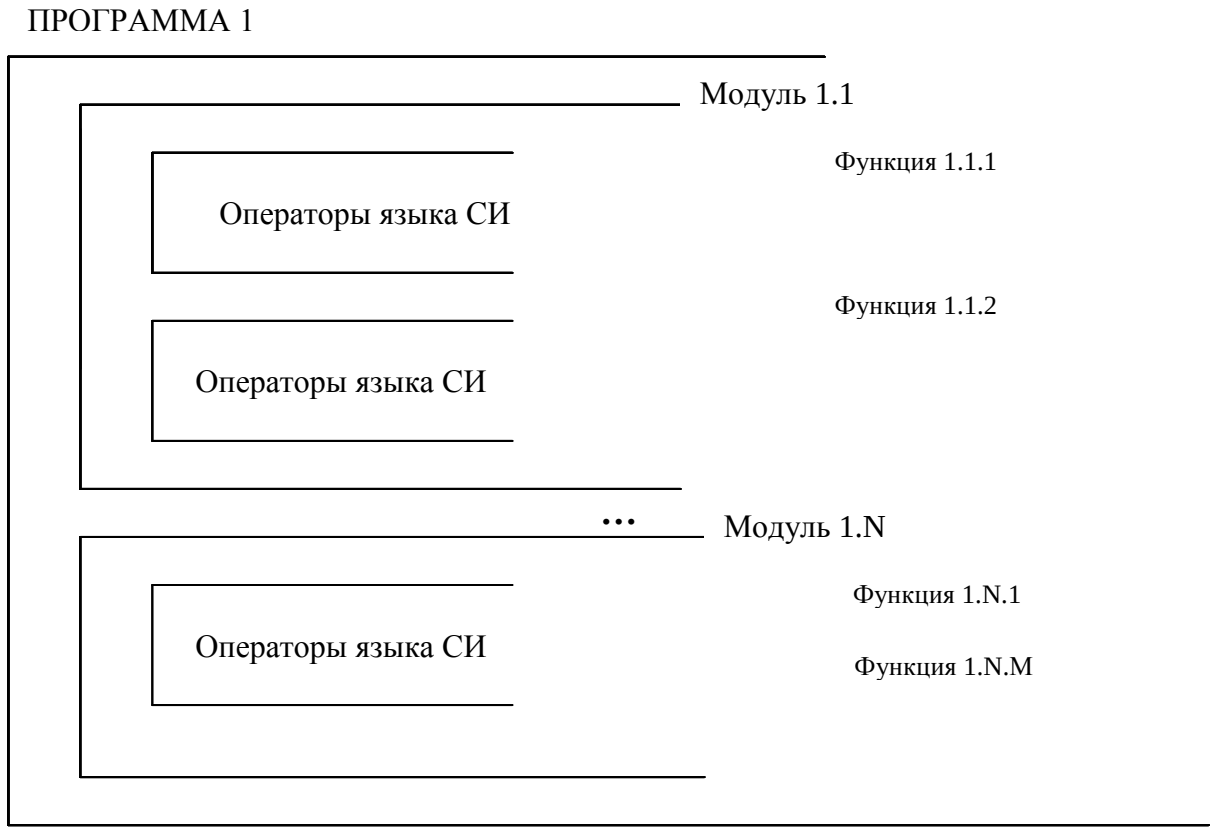


Рис. 1. Структура программы на языке Си

В отличие от программы на «Pascal», у программы на языке Си нет строки «заголовок» и строгого расположения разделов описания.

Принято начинать программу с комментариев, в которых указывается назначение программы, данные об исполнителе, могут быть перечислены требуемые внешние файлы. Текст программы также, должен быть разумно комментирован.

Например: /* Программа нахождения максимального элемента массива */ //Программа написана студентом гр. ИВТ – 61д И.С.Сидоровым

Строгого порядка описаний в языке Си нет, но действует правило, чтобы все объекты (константы, типы, переменные, функции и т.п.) были описаны до их применения (обращения к ним, действий над ними). Например, до операций ввода-вывода необходимо включение внешнего файла – stdio.h, в котором содержатся данные функции.

Значит, (модуль) программа может быть продолжена директивой включения внешних модулей:

include <имя_файла>

Например:

include < stdio.h>

подключить модуль функций стандартного ввода-вывода

и, возможно,

include <math.h>

модуль математических функций.

Директива # include служит командой для одной из частей компилятора, называемой *препроцессором.*

Дальше могут идти другие строки и функции препроцессора: описание констант:

define < имя> < значение>

Например:

define ConStr ‘ \0 ’

define Sto 100

Затем может быть описано определение нестандартного типа данных (тип пользователя):

typedef struct my {……}…;

и глобальные переменные (см. ниже, определяются как и локальные внутри функции).

Дальше указывается стандартное имя обязательной головной функции, возможно с входными и выходным параметрами:

[тип_выходного_параметра] main ([описание_входных_данных])

Как головная, так и любая другая функция состоит из блока локальных определений и операторов, который заключается в фигурные скобки, определяя его начало и канец данного блока: { }

Первая программа «Hello, World»

Эта программа, появившаяся в первом издании книги «Язык программирования Си» Кернигана и Ритчи, обычно является первой программой большинства учебников Си. Она печатает сообщение «Hello, world!» на стандартном устройстве вывода (которым, как правило, является монитор, но может быть и файл, какое-либо устройство или область в памяти, в зависимости от того, как отражается стандартное устройство вывода на данной платформе).

```
main() { printf("Hello, World!\n"); }
```

Несмотря на то, что на большинстве современных компиляторов эта программа может быть скомпилирована, она порождает несколько предупреждений на компиляторах стандарта ANSI C. Кроме того, этот код не будет компилироваться, если компилятор следует стандарту C99, так как в этом случае тип int больше не подразумевается для случаев, когда тип в результате функции не указан (а оформление функции main вообще описано отдельно). Эти сообщения можно убрать, если внести в эту программу несколько небольших изменений:

```
#include <stdio.h>
int main(void)
{    printf("Hello, World!\n");
   return 0;
}
```

В первой строке программы расположена директива препроцессора #include, встретив которую, компилятор её заменяет на полный текст файла, на который она ссылается. В данном случае эта строка будет заменена стандартным заголовочным файлом stdio.h. Угловые скобки указывают компилятору искать файл stdio.h в каталоге стандартных заголовочных файлов. Следующая строка является объявлением функции с именем main. Эта функция в программе Си является особенной, так как *выполняется первой* из написанных программистом при запуске программы. Слово int говорит, что функция main возвращает целое число. Фигурные скобки после функции main обозначают её определение. Следующая строка «вызывает» функцию printf. Включаемый заголовочный файл stdio.h содержит информацию, описывающую то, как нужно вызывать эту функцию. В данном примере этой функции передаётся единственный аргумент, содержащий текстовую строку "Hello, World!\n. Последовательность \n транслируется в символ «новая строка», который при отображении соответственно обозначает разрыв строки. Функция printf возвращает значение типа int, которое возвращает число напечатанных символов (в этом примере возвращаемое значение игнорируется) Выражение return заставляет программу прекратить выполнение функции (main в этом случае), возвращая вызвавшей функции значение, указанное после ключевого слова return (здесь 0). Так как текущая функция - это main, то вызывающим является код, который и запустил программу. Последняя фигурная скобка обозначает конец определения функции main. По стандарту C99, return 0 в main не обязательно (отсутствие return в main означает return 0;). В результате работы данной программы можно будет наблюдать следующий вид экрана: *Hello, World!*

Другой пример программы. В программе введенная пользователем строчка переводится в верхний регистр, то есть маленькие буквы становятся заглавными. Заглавные буквы и символы, не являющиеся латинскими буквами, не меняются. Обратите внимание на то, что с символами (переменными типа char) можно оперировать как с числами. В частности, 'Z' -'A' есть число, равное разности ASCII кодов символов 'A' и 'Z', то есть 26 — число букв в латинском алфавите. Символы

можно также сравнивать друг с другом, при этом сравниваются соответствующие им ASCII коды. Программа может иметь вид:

```
 #include <stdio.h>
 #define N 100
int main()
{ char a[N];
   int i;
   scanf ("%s", a);
   for(i = 0; a[i] != '\0'; i++)
     if( a[i] <= 'z') && (a[i] >= 'a')     a[i] += 'A' - 'a';
   printf ("%s\r\n", a);
      return  0;   }
```

ТИПЫ ДАННЫХ. ОПЕРАЦИИ И ОПЕРАТОРЫ

Система типов в Си подобна типам в других потомках Алгола, таких, как Паскаль. В Си имеются типы *целых чисел* различных размеров (short int, long int), со знаком (signed) и без (unsigned), чисел *с плавающей запятой* (float, double), *символов*, *перечисляемых типов* (enum) и *записей-структур* (struct). Кроме этого, язык Си предлагает *тип объединение* (union), с помощью которого можно либо хранить в одном месте памяти разнородные данные, не пересекающиеся по времени существования (это позволяет экономить память), либо обращаться к содержимому участка памяти, как к данным разных типов (что позволяет менять тип-интерпретацию данных, не меняя сами данные).

В языке возможно преобразование типов, но оно не всегда происходит автоматически. Только некоторые типы числовых данных полностью совместимы друг с другом. При таком преобразовании компилятор может выдать только предупреждение.

ХРАНЕНИЕ ДАННЫХ

Одной из самых важных функций любого языка программирования является предоставление возможностей для управления памятью и объектами, хранящимися в ней.

В Си есть три разных способа выделения памяти (классы памяти) для объектов:

- *Статическое выделение памяти*: пространство для объектов создаётся в сегменте данных программы в момент компиляции; время жизни таких объектов совпадает со временем жизни этого кода. Изменение таких объектов ведёт к так называемому в стандарте «неопределённому поведению» (англ. *undefined behaviour*). На практике эта операция приводит к ошибке времени выполнения.
- *Автоматическое выделение памяти*: объекты можно хранить в стеке; эта память затем автоматически освобождается и может быть использована снова, после того как программа выходит из блока, использующего его.
- *Динамическое выделение памяти*: блоки памяти нужного размера могут запрашиваться во время выполнения программы с помощью библиотечных функций malloc, realloc, calloc из области памяти, называемой кучей. Эти блоки освобождаются и могут быть использованы снова после вызова для них функции free.

Все три этих способа хранения данных пригодны в различных ситуациях и имеют свои преимущества и недостатки. Например, статическое выделение памяти не имеет накладных расходов по выделению, автоматическое выделение - лишь малые расходы при выделении, а вот динамическое выделение потенциально требует больших расходов и на выделение, и на освобождение памяти. С другой стороны, память стека гораздо больше ограничена, чем статическая или память в куче. Только динамическая память может использоваться в случаях, когда размер используемых объектов заранее неизвестен. Большинство программ на Си интенсивно используют все три этих способа.

Там, где это возможно, предпочтительным является автоматическое или статическое выделение памяти: такой способ хранения объектов управляется компилятором, что освобождает программиста от трудностей ручного выделения и освобождения памяти, как правило, служащего источником трудно отыскиваемых ошибок утечек памяти и повторного освобождения в программе. К сожалению, многие структуры данных имеют переменный размер во время выполнения

программы, поэтому из-за того, что автоматически и статически выделенные области должны иметь известный фиксированный размер во время компиляции, очень часто требуется использовать динамическое выделение. Массивы переменного размера - самый распространённый пример такого использования памяти.

Разнообразные вычисления — моделирование, решение алгебраических и дифференциальных уравнений — это то, для чего и создавались первые компьютеры. Давайте и мы научимся использовать компьютер для вычислений. Начнём со сложения двух чисел.

В нашей программе будут две целочисленные переменные: a и b. Это две ячейки памяти, в которых могут храниться целые числа из определенного диапазона значений (в 32-разрядной архитектуре от -2^{31}до $2^{31}-1$).

Переменные объявляются в начале тела функции main — после открывающей фигурной скобки. Объявление начинается со слова, обозначающего тип переменных, имена которых перечисляются через запятую после обозначения типа.

```
int a, b;
```

В языке Си есть несколько типов данных. Они делятся на две группы: *целые* типы:

char,

short,

int,

long,

unsigned char,

unsigned int,

unsigned long .

типы с *плавающей точкой* float, double.

Вот текст программы, складывающей два введенных целых числа:

```
#include <stdio.h>
int main () {
```

```
int a, b;
printf ("Введите два числа: \n");
scanf ("%d%d", &a, &b);
printf ("%d\n", a + b);
return 0;}
```

Функция `scanf`, также как и `printf`, определена в библиотеке `stdio`. Эта функция считывает данные, которые пользователь (тот, кто запустит вашу программу) вводит с клавиатуры. Слово `scan` означает «считывать данные», а `print` — «печатать данные». Буква «f» в конце соответствует первой букве английского слова «formated», то есть `scanf` и `printf` есть функции для форматированного ввода и вывода данных.

Первый аргумент у функции `scanf` — это `"%d%d"` (то, что стоит между открывающей скобкой и первой запятой). Первый аргумент является описанием формата входных данных, то есть описание типа данных, которые (как мы ожидаем) введёт пользователь. В этой программе мы ожидаем, что пользователь введет два целых числа.

Символ % служебный, с него начинается описание формата. Обычно после него идет один или два символа, определяющих тип входных данных. Формат "%d" соответствует целому числу в десятичной системе счисления (decimal integer). Если вы напишете "%x", то функция будет ожидать ввода целого числа, записанного в шестнадцатеричной системе счисления.

ПРИОРИТЕТ ОПЕРАЦИЙ

Операции языка Си, в порядке снижения приоритета:

Лексемы	Операция	Класс	Приоритет	Ассоциативность
имена, литералы	простые лексемы	первичный	16	нет
`a[k]`	Индексы	постфиксный	16	слева направо
`f(…)`	вызов функции	постфиксный	16	слева направо
`.`	прямой выбор	постфиксный	16	слева направо
`->`	опосредованный выбор	постфиксный	16	слева направо

++ --	положительное и отрицательное приращение	префиксный	16	слева направо
(имя типа) {init}	составной литерал (C99)	постфиксный	16	слева направо
++ --	положительное и отрицательное приращение	постфиксный	15	справа налево
Sizeof	Размер	Унарный	15	справа налево
~	побитовое НЕ	Унарный	15	справа налево
!	логическое НЕ	Унарный	15	справа налево
- +	изменение знака, плюс	Унарный	15	справа налево
&	Адрес	Унарный	15	справа налево
*	опосредование (разыменование)	Унарный	15	справа налево
(имя типа)	приведение типа	Унарный	14	справа налево
* / %	мультипликативные операции	Бинарный	13	слева направо
+ -	аддитивные операции	Бинарный	12	слева направо
<< >>	сдвиг влево и вправо	Бинарный	11	слева направо
< > <= >=	Отношения	Бинарный	10	слева направо
== !=	равенство/неравенство	Бинарный	9	слева направо
&	побитовое И	Бинарный	8	слева направо
^	побитовое исключающее ИЛИ	Бинарный	7	слева направо
|	побитовое ИЛИ	Бинарный	6	слева направо
&&	логическое И	Бинарный	5	слева направо
||	логическое ИЛИ	Бинарный	4	слева направо
? :	Условие	тернарный	3	справа налево
= += -= *= /= %= <<= >>= &= ^= |=	Присваивание	Бинарный	2	справа налево
,	последовательное вычисление	Бинарный	1	слева направо

Многие элементы Си потенциально опасны, а последствия неправильного использования этих элементов зачастую непредсказуемы. В связи со сравнительно низким уровнем языка многие случаи неправильного использования опасных элементов не обнаруживаются и не могут быть обнаружены ни при компиляции, ни во время исполнения. Это часто приводит к непредсказуемому поведению программы. Иногда в результате неграмотного использования элементов языка

появляются уязвимости в системе безопасности. Необходимо заметить, что использования многих таких элементов можно избежать.

Примером ошибки является *обращение к несуществующему элементу* массива. Несмотря на то, что Си непосредственно поддерживает статические массивы, он не имеет средств проверки индексов массивов (проверки границ). Например, возможна запись в шестой элемент массива из пяти элементов, что, естественно, приведёт к непредсказуемым результатам. Частный случай такой ошибки называется ошибкой переполнения буфера. Ошибки такого рода приводят к большинству проблем с безопасностью.

Другим источником опасных ситуаций служит механизм указателей. Указатель может ссылаться на любой объект в памяти, включая и исполняемый код программы, и неправильное использование указателей может порождать непредсказуемые эффекты и приводить к катастрофичным последствиям. Например, указатель может быть неинициализированным или, в результате неверных арифметических операций над указателем, указывать в произвольное место памяти; на некоторых платформах работа с таким указателем может вызвать аппаратную остановку программы, на незащищённых же платформах это может привести к порче произвольных данных в памяти, причём эта порча может проявиться в самые произвольные моменты времени и намного позже момента порчи. Также область динамической памяти, на которую ссылается указатель, может быть освобождена (и даже выделена после этого под другой объект) - такие указатели называются «висячими». Или, наоборот, после манипуляций с указателями на область динамической памяти может не остаться ссылок, и тогда эта область, называемая «мусором», никогда не будет освобождена, что может приводить к «утечкам памяти» в программе.

Проблемой является также то, что автоматически и динамически создаваемые объекты не инициализируются и они могут содержать значения, оставшиеся в памяти от ранее удалённых объектов. Такое значение полностью непредсказуемо, оно меняется от одной машины к другой, от запуска к запуску, от вызова функции

к вызову. Если программа использует такое значение, то результат будет непредсказуемым и не обязательно проявится сразу.

Ещё одной распространённой проблемой является то, что память не может быть использована снова, пока она не будет освобождена программистом с помощью функции free(). В результате программист может случайно забыть освобождать эту память, но продолжать её выделять, занимая всё большее и большее пространство. Это обозначается термином утечка памяти. Наоборот, возможно освободить память слишком рано, но продолжать её использовать. Из-за того, что система выделения может использовать освобождённую память по-другому, это ведёт к непредсказуемым последствиям. Эти проблемы решаются в языках со сборкой мусора. С другой стороны, если память выделяется в функции и должна освобождаться после выхода из функции, данная проблема решается с помощью автоматического вызова деструкторов в языке C++ или с помощью локальных массивов, с использованием расширения C99.

Функции с переменным количеством аргументов также являются потенциальным источником проблем. В отличие от обычных функций, имеющих прототип, стандартом не регламентируется проверка функций с переменным числом аргументов. Если передаётся неправильный тип данных, то возникает непредсказуемый, если не фатальный результат. Например, семейство функций printf стандартной библиотеки языка Си, используемое для генерации форматированного текста для вывода, хорошо известно за его потенциально опасный интерфейс с переменным числом аргументов, которые описываются строкой формата. Проверка типов в функциях с переменным числом аргументов является задачей каждой конкретной реализации такой функции, но многие современные компиляторы, в частности, проверяют типы в каждом вызове printf, генерируя предупреждения в случаях, когда список аргументов не соответствует строке формата. Следует заметить, что невозможно статически проконтролировать даже все вызовы функции printf, поскольку строка формата может создаваться в программе динамически, поэтому, как правило, никаких проверок других функций с переменным числом аргументов компилятором не производится. Отметим, что

для каждого типа данных существует несколько форматов, и наоборот, для разных типов можно использовать один и тот же формат. Рассмотренная программа умеет складывать только целые числа. Если вы хотите складывать действительные числа, то эту программу нужно несколько модифицировать. Ниже приведена программа, которая считывает два действительных числа и выводит результат четырех арифметических операций: сложения, вычитания, умножения и деления. Причём программа выводит результаты вычислений два раза — сначала в обычном виде, а потом со специальным форматированием. Формат `"%10.3lf"` соответствует выводу числа типа `double`, при котором под запись числа выделяется ровно 10 позиций (если это возможно), а после запятой пишется ровно три знака. Равнение происходит по правому краю.

```
/* Программа "Арифметические операции с числами с плавающей точкой" */
#include <stdio.h>
int main ()
{    double a, b;
   printf ("Введите два числа: ");
   while(scanf ("%lf%lf", &a, &b) == 2 )
          { printf ("%lf %lf %lf %lf\n", a + b, a - b, a * b, a / b );
            printf ("a + b=%10.3lf\n a - b=%10.3lf\n a * b=%10.3lf\n a / b=%10.3lf\n",
              a + b, a - b, a * b, a / b );  }
   return 0;
}
```

В этой программе мы встречаемся с оператором `while`. Конструкция while (A) B; означает буквально следующее: «Пока выполнено условие `A`, делать `B`.» или, другими словами, «Выполнять в цикле `B`, проверяя перед каждой итерацией, что выполнено условие `A`». В нашем случае `A` есть scanf("%lf%lf", &a, &b) == 2. Что соответствует логическому выражению: «пользователь ввёл два действительных числа, и они удачно считаны в переменные a и b». Т.о., эта программа будет считывать пары чисел и выводить результаты арифметических операций, пока пользователь не введёт что-нибудь не похожее на число. Цикл while закончится тогда, когда функция scanf не сможет успешно считать два числа.

Заметьте, что после каждой команды стоит точка с запятой. Одна из самых популярных синтаксических ошибок начинающих программистов — это не ставить точку с запятой в конце команды.

Задача: написать программу, которая находит максимум из наперёд заданного количества введенных целых чисел. На ввод поступает количество чисел, а затем сами числа.

```
/* Программа max.c "Максимум чисел" */
 #include <stdio.h>
 int main () {
     int i, n, a, max;
     printf ("Введите количество чисел: ");
     scanf ("%d", &n);
     printf ("Введите %d чисел через пробел: ", n);
/* предположим, что первое, введённое – максимальное */
     scanf ("%d", &max);
    for(i = 1; i < n ; i++) { scanf ("%d", &a);
                              if(a > max) max = a;}
     printf ("%d", max);      return 0; }
```

Числа можно вводить, разделяя их пробелом SPACE, символом табуляции TAB или нажимая после каждого введенного числа ENTER. Символы SPACE, TAB, ENTER называются пробельными символами (white space). Функция scanf считывает объекты, разделенные любым числом пробельных символов.

Обратите внимание на второй printf — он имеет два аргумента:

```
printf ("Введите %d чисел: ", n);
```

Первый аргумент — это строчка `"Введите %d чисел: "`, которая задает формат того, что будет печататься. В этой строчке встречается выражение `%d`, которое соответствует формату «десятичная запись целого числа».

На месте этого выражения будет напечатано на экране компьютера значение второго аргумента, то есть значение переменной `n`.

В этой программе впервые встречается с *условный оператор* `if` и *оператор цикла* `for`.

Оператор условного перехода записывается так: `if(A) B;`и соответствует предложению: «Если выполнено условие A, то сделать B.».

Оператор for устроен следующим образом: `for(A; B; C ) D;`

Элемент `D` может быть как одной командой, так и произвольным набором команд, заключенных в *блок*. Команды объединяются в блок с помощью заключения их в фигурные скобки. В нашем случае `D` это

```
{ scanf ("%d", &a);
 if(a > max) max = a;}
```

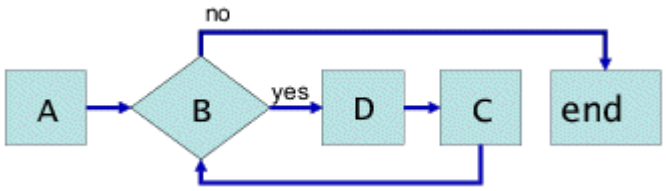
Элемент `D` называется телом цикла — это то, что будет выполняться несколько раз. Сколько именно? Это зависит от `B` — тело `D` будет выполняться, пока выполнено условие `B`.

Например, строчка `for(i = 0 ; i < 10 ; i++) { printf ("Hi!"); }`

означает 10 раз напечатать выражение `Hi!`. Т.е.:

1. Положим `i = 0`;
2. Проверим, что `i < 10`; если условие не выполнено, то перейдем к пункту 5;
3. Выполним `printf ("Hi!");`
4. Выполним `i++` (увеличим переменную `i` на 1) и перейдем к пункту 2;
5. Конец цикла;

Логику оператора `for` можно изобразить в виде блок-схемы:



Логика программы следующая:

1. Объявляем все переменные, которые встретятся в нашей программе:
   - `n` — количество чисел,
   - `i` — переменная цикла,
   - `max` — переменная для хранения текущего найденного максимума,
   - `a` — переменная с очередным числом, в неё мы считываем каждое число.

2. Печатаем на экран приглашение ввести число n.
3. Считываем n.
4. Печатаем на экран приглашение «Введите n чисел».
5. Считываем первое из этих n чисел в переменную max.
6. В цикле n - 1 раз считываем очередное число, и если оно больше, чем текущий максимум, то обновляем переменную max.
7. Выводим значение переменной max .

Вы заметили, что когда в scanf указываем переменные, в которые нужно поместить считываемые данные, то перед именем переменной ставим знак & (амперсанд), в то время как в printf при печати значений переменных амперсанд не ставится.

Операция «амперсанд» — это операция взятия адреса. Если a есть значение переменной a, то &a — это адрес в памяти компьютера, где хранится переменная a: одно есть само число, другое — место, где это число хранится.

Функции scanf нужно знать не текущее значение переменной, а её адрес, чтобы положить туда результат считывания. Функции printf нужно знать значения переменных, чтобы их печатать. Функции scanf нужно передавать адреса переменных, куда помещать считываемые данные.

На примере задачи «Таблица умножения» рассмотрим применение вложенности циклов. Задача: вывести на экран компьютера таблицу умножения размера n x n, где n вводит пользователь с клавиатуры. Возможный вариант:

```
/* Программа table.c "Таблица умножения" */
#include <stdio.h>
int main()
{ int i, j, n;
    printf ("Введите n: ");
    scanf ("%d", &n);
    for(i = 1; i <= n ; i++) {
        for(j = 1; j <= n ; j++) {printf ("%5d", i * j);}
        printf("\n");         }
```

```
    return 0;}
```

Переменные `i` и `j` соответствуют номеру строчки и номеру столбца. Переменная внешнего цикла `i` сначала равна 1. Начинает работу внутренний цикл:

```
for(j = 1; j <= n ; j++)
                { printf ("%5d", i * j); }
```

— он печатает первую строчку таблицы.

Переменная внутреннего цикла `j` меняется от 1 до `n` включительно. При этом для каждого значения `j` выводится на экран результат произведения `i*j`.

Формат вывода определяется как `"%5d"`, что означает "печатать целое число, выделяя под него 5 позиций". При этом если в числе меньше, чем пять разрядов, то оно придвигается к правому краю, а слева добавляются пробелы.

Когда заканчивается внутренний цикл, происходит переход на новую строчку (команда - printf ("\n");). После этого переменная внешнего цикла `i` увеличивает своё значение на 1 и становится равна 2. Затем снова запускается внутренний цикл, и печатается вторая строчка таблицы умножения.

Результат работы программы (пользователь ввёл число 5):

```
Введите n: 5
    1    2    3    4    5
    2    4    6    8   10
    3    6    9   12   15
    4    8   12   16   20
    5   10   15   20   25
```

Удаление символов из строки демонстрирует функция squeeze, которая удаляет все символы `c` из строки `s` по алгоритму:

- пока не будет достигнут конец строки,
- если очередной символ строки не равен символу для удаления,
- записать очередной символ по месту индекса `j`, после этого последний увеличить на единицу.

Алгоритм сводится к тому, что символ по индексу `i` затирается следующим символом, если он совпал с символом для удаления; т.к. запись идет по счетчику `j`, а он увеличивается лишь тогда, когда символы из строки и для удаления не совпадают.

Алгоритм удаления тех символов строки, которые встречаются в другой строке, отличается от предыдущего тем, что нужно по очереди извлекать символы из второй строки и удалять их из первой. Т.е. приведенный выше алгоритм следует вложить в цикл перебора символов второй строки.

Удаление всех символов «c» из строки

```
#include <stdio.h>
#define MAX 100
void squeeze (char s[], int c);
main () { char str0[MAX];
         int i, c;
   for (i = 0; (c = getchar()) != '\n'; i++) str0[i] = c;
   str0[i] = '\0';
   c = getchar();
   squeeze (str0, c);
   printf("%s\n", str0);}
void squeeze (char s[], int c) {
   int i, j;
   for (i = j = 0; s[i] != '\0'; i++) if (s[i] != c) s[j++] =
s[i];
   s[j] = '\0';                  }
```

Удаление всех символов, встречающихся в строке s2, из строки s1

```
#include <stdio.h>
#define MAX 100
#define DEL 10
void squeeze (char s[], char s1[]);
void written (char s[]);
main()                 {
   char str[MAX];
   char str1[DEL];
   written (str);
   written (str1);
   squeeze (str, str1);
   printf("%s\n", str); }
void written (char s[100]) {
   int i, c;
   i = 0;
   while ((c = getchar()) != '\n') {
         s[i] = c;
         ++i;
                                  }
   s[i] = '\0';
                           }
void squeeze (char s[], char s2[])    {
   int k, i, j;
   for (k = 0; s2[k] != '\0'; k++) {
         for (i = j = 0; s[i] != '\0'; i++)
               if (s[i] != s2[k]) s[j++] = s[i];
         s[j] = '\0';              }   }
```

Примечание. Во второй программе запись символов в строку выделена в отдельную функцию, т.к. надо записать две строки. Иначе пришлось бы дублировать код.

ВИДИМОСТЬ ПЕРЕМЕННЫХ

Переменную j можно было объявить внутри первого цикла:

```
/* Программа table.c "Таблица умножения"  */
#include <stdio.h>
int main() {
    int i, n;
      printf ("Введите n: ");
      scanf ("%d", &n);
      for(i = 1; i <= n ; i++)
          {int j;      // Начало области видимости переменной  j
             for(j = 1; j <= n ; j++)
                    {printf ("%5d", i * j);}
             printf("\n");
          }           // Конец области видимости переменной  j
    return 0;}
```

Как только вы начинаете новый блок (открываете фигурную скобку), вы можете объявить новые переменные. Эти переменные будут видны только внутри данного блока, и их называют локальными переменными блока. При этом вполне допустима ситуация, когда вы объявляете переменную, имя которой совпадает с одной из переменной, объявленной снаружи. В этом случае внутри блока данное имя связывается с локальной переменной, и операции с этой переменной никак не влияют на состояние внешней переменной:

```
int main() {
    int a = 1;   //Внешняя (по отношению к следующему блоку) переменная
    { // Новый блок можно начинать в любом месте
              int a;   // Локальная переменная блока
       a = 2;
       printf("inside %d\n", a);
            }     // Конец видимости локальной переменной a
    printf("outside %d\n", a); // Будет напечатано неизменившееся значение внешней переменной, то есть 1;
    return 0;
           }
```

СИМВОЛЫ В СИ

В языке С есть тип `char` для символов. Каждому символу сопоставлено число от 0 до 255, которое называется ASCII-кодом символа. Например, символу 'A' латинская соответствует число 65. Символами можно оперировать, как числами, и, наоборот, переменные типа `int` можно интепретировать как символы (сравнивать с символами или печатать как символы).

Чтобы считывать один символ, есть функция `getchar` из библиотеки `stdio`. В следующей программе считывается символ и печатается в двух форматах: как символ (формат `"%c"`) и как число (формат `"%d"`). Это делается до тех пор, пока символ (а точнее, его ASCII код) не будет равен 27, то есть пока не будет нажата клавиша ESC.

Выражение `ch!= 27` означает логическое `ch` $\neq 27$.

```
#include <stdio.h>
int main ()
     {int ch;
       do { ch = getchar();
            printf ("Вы нажали %c. ASCII код = %d\n", ch, ch);
          } while (ch != 27);
       return(0);
     }
```

Подсчет количества символов, строк и слов во введённом тексте

```
#include <stdio.h>
main() {
   char ch, flag=-1;
   unsigned c=0, n=0, w=0;
   while ((ch = getchar()) != EOF) {if (ch == '\n') n++; // строки
                                       else c++; // символы
         if (ch == ' ' || ch == '\n') flag = -1; // слова
         else if (flag == -1) {flag = 1; w++;}
                                  }
   printf("%u %u %u\n", c, n, w);
       }
```

Задача нахождения факториала

Факториал числа n, это произведение первых n натуральных чисел:

$n! = 1 \cdot 2 \cdot \ldots \cdot n.$

Ниже приведена программа, содержащая определение функции factorial, которая вычисляет факториал.

```
#include <stdio.h>
  int factorial(int x) {
         return !x ? 1 : x * factorial(x - 1);
                        }
void main() {
      int n;
        while( scanf("%d", &n) == 1)
           printf("%d\n", factorial (n));
             }
```

Это определение основано на следующей **рекуррентной** формуле:

$n! = n \cdot (n-1)!, \quad 0! = 1.$

Для вычисления факторила `n!` эта функция вызывает самоё себя с аргументом `n-1`

Если бы не было строчки: if(n == 0) return 1;

то функция `factorial` постоянно бы вызывала самоё себя, и во время выполнения программы мы получили бы сообщение об ошибке «переполнение стека» или «превышена глубина рекурсии». Для того, чтобы этого не было, необходимо, чтобы при некоторых значения аргумента функция не вызывала самоё себя, а вычисляла свое значение «самостоятельно».

Идея рекурсии заключается в сведении задачи к этой же задаче, но для более простых случаев (например, для меньшего значения аргумента `n`). Процесс сведения задачи к предыдущей должен когда-нибудь заканчиваться, поэтому рекурсивные функции для простейших входных данных должны «знать», чему они равны, и не делать рекурсивных вызовов.

УКАЗАТЕЛИ И МАССИВЫ

Пример поможет закрепить понимание, как работать с указателем на массив. В выводе программы обратите внимание, адреса различаются между собой на 4 байта, т.е. переменная типа int занимает 4 байта. Также на примере можно удостовериться, что элементы массива располагаются в памяти последовательно друг за другом.

& - операция взятия адреса переменной; * - операция извлечения значения по указанному адресу; применяется к переменным, содержащим адрес (т.е. к указателям).

Когда переменная-указатель объявляется (например, int *q), перед ее именем также ставится знак *. Однако в данном контексте он просто сообщает, что данная переменная - это указатель, и знак * не имеет отношения к извлечению значения.

#include <stdio.h>

```
main () {
   // объявление массива и его инициализация
   int arr[10] = {101, 102, 103, 104, 105, 106, 107, 108, 109, 110};
   int *p, *q; // указатели на тип int
   int i;
   printf("Массив: ");
   for (i = 0; i < 10; i++)
          printf(" [%d]=%d|", i, arr[i]);
   printf("\n\n");
// 1 ---------------------------------------------------------------------
   p = &arr[0];
   printf("Адрес 1-го элемента массива (p): %p\n", p);
   printf("Значение 1-го элемента массива (*p): %d\n", *p);
   q = p + 1;
   printf("Адрес 2-го элемента массива (q): %p\n", q);
   printf("Значение 2-го элемента массива (*q): %d\n", *q);
   q = q + 2;
   printf("Адрес 4-го элемента массива (q): %p\n", q);
   printf("Значение 4-го элемента массива (*q): %d\n", *q);
   printf("\n");
// 2 ---------------------------------------------------------------------
   // имя массива является указателем на его первый элемент
   printf("Адрес 1-го элемента массива (arr): %p\n", arr);
```

```
    printf("Значение 1-го элемента массива (*arr): %d\n", *arr);
    // Поэтому ...
    p = arr + 2; // то же самое, что p = &arr[0] + 2
    printf("Адрес 3-го элемента массива (arr + 2): %p\n", p);
    printf("Значение 3-го элемента массива (*(arr+2)): %d\n", *p);
    printf("Адрес 6-го элемента массива (p + 3): %p\n", p+3);
    printf("Значение 6-го элемента массива (*(p+3)): %d\n", *(p+3));
    printf("\n");
// 3 ------------------------------------------------------------------
    // указатель можно употреблять в выражениях с индексом
    p = arr;
    printf("Значение 10-го элемента массива (p[9]): %d\n", p[9]);
    printf("\n");
// 4 ------------------------------------------------------------------
    p = arr; // arr++ использовать нельзя, а p++ можно
    for (i = 0; i < 10; i++, p++)
            printf("%2d-й: %p    %d\n", i+1, p, *p);
}
```

В результате программа выводит:

```
Массив:  [0]=101| [1]=102| [2]=103| [3]=104| [4]=105| [5]=106| [6]=107|
[7]=108| [8]=109| [9]=110|
Адрес 1-го элемента массива (p): 0x7fff5c4c0d60
Значение 1-го элемента массива (*p): 101
Адрес 2-го элемента массива (q): 0x7fff5c4c0d64
Значение 2-го элемента массива (*q): 102
Адрес 4-го элемента массива (q): 0x7fff5c4c0d6c
Значение 4-го элемента массива (*q): 104

Адрес 1-го элемента массива (arr): 0x7fff5c4c0d60
Значение 1-го элемента массива (*arr): 101
Адрес 3-го элемента массива (arr + 2): 0x7fff5c4c0d68
Значение 3-го элемента массива (*(arr+2)): 103
Адрес 6-го элемента массива (p + 3): 0x7fff5c4c0d74
Значение 6-го элемента массива (*(p+3)): 106

Значение 10-го элемента массива (p[9]): 110

 1-й: 0x7fff5c4c0d60    101
 2-й: 0x7fff5c4c0d64    102
 3-й: 0x7fff5c4c0d68    103
 4-й: 0x7fff5c4c0d6c    104
 5-й: 0x7fff5c4c0d70    105
```

```
 6-й: 0x7fff5c4c0d74    106
 7-й: 0x7fff5c4c0d78    107
 8-й: 0x7fff5c4c0d7c    108
 9-й: 0x7fff5c4c0d80    109
10-й: 0x7fff5c4c0d84    11
```

СОРТИРОВКА

Задача «сортировки» (упорядочения) — одна из первых интересных и сложных задач теории алгоритмов. Рассмотрим способы упорядочения посредством языка Си.

Метод «пузырька»

Один из простейших алгоритмов решения — «метод пузырька».

```
#include<stdio.h>
#define N 1000
int main() {
   int n, i, j;
   int a[N];
    scanf("%d", &n); // считываем количество чисел n
   // считываем n чисел
   for(i=0;i<n;i++) {scanf("%d",&a[i]);}
   for(i=0;i<n;i++) {
      // в цикле сравниваем два соседних элемента
      for(j=0;j<n-i-1;j++) {
          if(a[j] > a[j+1])   {
             // если они не упорядочены,то меняем их местами
             int tmp = a[j]; a[j]=a[j+1]; a[j+1]=tmp;
                                }
                            }
                        }
             }
```

После первого «пробега» самый большой элемент массива окажется на последнем месте (он и был тем самым «пузырьком», который «всплыл»). После второго цикла второй по величине элемент будет находиться на предпоследнем месте и т.д.

Рассмотрим программу упорядочения строк в алфавитном порядке

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#define N 100
#define M 30
int main(int argc, char* argv[]) {
      char a[N][M];
      int n, i;
       scanf("%d", &n);
       for (i=0; i<n; i++)
       scanf("%s", &a[i]);
    qsort(a, n, sizeof(char[M]), (int (*)(const void *,const
void *)) strcmp);
    for (i=0; i<n; i++) printf("%s\n", a[i]);
    return 0;                          }
```

Обратите внимание на сложное приведение типов.

Функция `strcmp`, объявленная в файле `string.h`, имеет следующий прототип: `int strcmp(const char*, const char*);`

Т.е. функция получает два аргумента - указатели на кусочки памяти, где хранятся элементы типа `char`, а именно: два массива символов, которые не могут быть изменены внутри функции `strcmp` (запрет на изменение задается с помощью модификатора `const`).

Одновременно, в качестве четвертого элемента функция `qsort` хотось бы иметь функцию типа `int cmp(const void*, const void*);`

В языке Си можно осуществлять приведение типов, являющихся типами функции. В данном примере тип

`int (*)(const char*, const char*);` // функция, получающая два элемента типа `'const char *'` и возвращающая элемент типа `'int'`,

приводится к типу

`int (*)(const void*, const void*);` // функция, получающая два элемента типа `'const void *'` и возвращающая элемент типа `'int'`.

Функция `strcmp` осуществляет сравнение двух строк и определяет, какая из двух строк идёт первой в алфавитном порядке (стравнивает две строки в лексографическом порядке), она возвращает:

1- если 1-я строка больше 2-й (идёт после второй в алфавитном порядке),

0 - если они совпадают,

1 - если 1-я меньше 2-й.

ДИНАМИЧЕСКОЕ ВЫДЕЛЕНИЕ ПАМЯТИ

Приведём ример программы, в которой память под массив выделяется динамически:

```
     #include <stdio.h>
     #include <stdlib.h>
     #include <malloc.h>
     #define N 1000
int cmp(const void *a, const void *b) {return *(int*)a-*(int*)b;}
     int main() {
        int n, i;
        int *a;
        scanf("%d", &n);
        a = (int*) malloc(sizeof(int)*n);
        for(i = 0 ; i < n; i++) {scanf("%d", &a[i]);}
        qsort(a, n, sizeof(int), cmp );
        for(i = 0 ; i < n; i++) {printf("%d ", a[i]);}
        free(a);
        return 0;}
```

Функция `malloc` (от англ. *memory allocation - выделение памяти*) делает запрос к ядру ОС по выделению заданного количества байт. Единственный аргумент этой функции — число байт, которое вам нужно. В качестве результата функция возвращает указатель на начало выделенной памяти. Указатель на начало выделенной памяти &mbsah - это адрес ячейки памяти, начиная с которого идут N байт, которые вы можете использовать. Всю память, которая была выделена с помощью функции `malloc`, нужно освобождать с помощью функции `free`. Аргумент функции `free` – это указатель на начало выделенной когда-то памяти.

Пример использования функции QSORT из STDLIB

Вход: В первой строчке дано N, а затем следует N строчек вида ИМЯ ТЕЛЕФОН

Выход: Выведите телефонный справочник в алфавитном порядке и в порядке возрастания телефонных номеров.

```
#include <stdio.h>
#include <stdlib.h>
typedef struct person_t {
    char * name;
    int phone;          } person_t;
int namecmp(const void* a, const void* b);
int phonecmp(const void* a, const void* b);
 int main(void)
 {  char name[1000];
    person_t *base;
    int n, i, phone;
    scanf("%d", &n);
    base = malloc(n * sizeof(person_t));
```

/* Считываем телефонную базу */

```
    for(i = 0 ; i < n ; i++) {
        scanf("%s%d", name, &phone);
```

/* Выделяем память под массив `base[i].name`, учитывая символ конца строки */

```
            base[i].name = malloc((strlen(name)+1) * sizeof(char));
            strcpy(base[i].name, name);
            base[i].phone = phone;}


    qsort(base, n, sizeof(person_t), namecmp);
```

/* Напечатаем в алфавитном порядке */

```
    for(i = 0 ; i < n ; i++){printf("%20s %d\n", base[i].name,
base[i].phone);}


    qsort(base, n, sizeof(person_t), phonecmp);
```

/* Напечатаем в порядке возрастания номеров*/

```
        for(i = 0 ; i < n ; i++) {printf("%20s %d\n", base[i].name,
base[i].phone);}
```

/* Освобождаем выделенную память */

```
        for(i = 0 ; i < n ; i++) {free(base[i].name);}
        free(base);
        return 0; }
    int namecmp(const void* a, const void* b)
    {     person_t *pa = a;
         person_t *pb = b;
         return strcmp(pa->name, pb->name);}
    int phonecmp(const void* a, const void* b)
    {     person_t *pa = a;
         person_t *pb = b;
         return pa->phone - pb->phone;}
```

ФАЙЛ НА ЯЗЫКЕ СИ

```
/*   file:  rpn.cc
   title: Tokenizer functions
            yylex    - функция разбиения входного потока на токены.
            yyerror  - функция обработки ошибок
            main     - главная функция      */
 #include <stdio.h>
 #include <ctype.h>
 #include "rpn.tab.hh"
int yyparse(void);
int yyerror (const char *s)  /* Called by yyparse on error */
{printf ("%s\n", s);}
int yylex (void){
     int c;
     while ((c = getchar ()) == ' ' || c == '\t');
     if (c == '.' || isdigit (c)){
         ungetc (c, stdin);
         scanf ("%lf", &yylval);
         return NUM;             }
      if (c == EOF) return 0;
```

```
      return c;  }
int main (void)
{return yyparse ();}
```

Команды компиляции

```
$ bison -d rpn.yy      # -> produce files rpn.tab.cc  and rpn.tab.hh
$ gcc rpn.tab.cc rpn.cc -lm -o rpn #  produce executable file rpn
```

СОЗДАНИЕ БИБЛИОТЕК

Библиотека complex для работы с комплексными числами.

Библиотека будет содержать два исходных файла: `complex.h` и `complex.c`

```
/* complex.h */
typedef struct { double a,b; } complex_t;
complex_t mul(complex_t x, complex_t y);
/*complex.c */
#include "complex.h"
complex_t mul(complex_t x, complex_t y)
 {   complex_t t;
     t.a = x.a * y.a - x.b * y.b;
     t.b = x.a * y.b + x.b * y.a;
     return t;}
```

Пример использования библиотеки:

- добавить в заголовок c/cpp-файла строчку `#include <complex.h>`
- организовать папки или прописать путь, чтобы компилятор находил `complex.h`
- чтобы сборщик (linker) находил файл библиотеки `complex`.

```
/*test_complex.c */
 #include <stdio.h>
 #include "complex.h"
int main()
{    complex_t x = {1,2};
     complex_t y = {3,4};
     complex_t z = mul(x,y);
     printf ("z = (%lf, %lf)\n", z.a, z.b ); return 0; }
```

Создание и использование библиотек в GNU C. Подключение библиотек в виде исходных

Есть простой способ подключения библиотеки функций прямо в виде исходников:

```
$ gcc test_complex.c complex.c -o test_complex
```

В итоге получается запускаемый файл `test_complex`.

Подключение библиотеки в виде динамически загружаемой библиотеки

```
$ gcc -shared complex.c -o libcomplex.so      #
```

создание динамически загружаемой библиотеки

```
$ gcc test_complex.c -L. -lcomplex -o test_complex #
```

компиляция тестовой программы

```
$ export LD_LIBRARY_PATH=.; ./test_complex         #
```

запуск тестовой программы

- Есть ряд стандартных путей (), где система ищет динамически загружаемые библиотеки. Текущая директория по умолчанию не является таким местом, поэтому нужно явно указать, что библиотеки могут лежать также и в текущей директории. Переменная `LD_LIBRARY_PATH` содержит дополнительный список путей (разделённых двоеточием), где система будет искать библиотеки.
- Расширение `.so` соответствует `shared objects`, то есть библиотека объектов общего доступа.

Подключение библиотеки в виде статически загружаемой библиотеки

Для компилятора Gnu C команды создания статически подключаемой библиотеки и подключения её к проекту выглядят следующим образом:

```
$ gcc -c complex.c -o libcomplex.a                 #
```

создание статически подключаемой библиотеки

```
$ gcc test_complex.c -L. -lcomplex -o test_complex #
```

компиляция тестовой программы

```
$ ./test_complex                                   #
```

запуск тестовой программы

ГРАФИКА В СИ

В компании Borland сделали одну графическую библиотеку. Теперь она часть языка СИ. Паскаль всегда рассматривается комплексно, с графикой. В СИ - язык отдельно, графика отдельно. Есть книги по языку, есть книги по графике. Для работы Вашей программы с графикой, придётся комплектовать её файлом

EGAVGA.BGI, который должен быть размещён в той же папке, что и программа, или в подпапке, чтобы случайно не стёрли. Также надо разрешить компилятору работать с графикой. Выставьте флажок в меню: options->linker->libraries->Graphic library. Освоив низкоуровнеувую графику, Вы без труда освоите графику под Windows. Принципиальных отличий нет, если не считать некоторых изменений в именах функций и того, что эти функции могут быть членами класса (Visual C++). Главное - понять концепцию.

Самый распространённый способ - использовать функцию *initgraph*, которая задаёт и режим и резрешение и указывает путь к EGAVGA.BGI. Рассмотрим её синтаксис: initgraph(*int far grdriver, *int far grmode, char *path);

Первый аргумент указывает тип видеоадаптера. Он принимает числовое значение: Аргумент grdriver.

Второй аргумент - графический режим, то есть разрешение (низкое, среднее, высокое). Иногда ещё режим называют модой.

Третий аргумент - *путь* к файлу драйвера EGAVGA.BGI. Если этот файл в папке c:\tc\bgi\, то значение его будет таково: c:\\tc\\bgi\\. Т.к. одинарный слэш (\) уже зарезервирован, в Си все пути указываются с двойными слэшами, но если вы прописываете путь к библиотечному файлу, то кавычки будут одинарными: #include "c:\tc\bin\my\biblioteka.h" Итак, всё готово, чтобы перевести компьютер в графический режим.

```
#include <conio.h>
#include<graphics.h>
void main() { clrscr();
cprintf("Текстовый режим."); getch();
//Переходим в графический режим!
int a=9, b=2;
initgraph(&a, &b, "c:\\tc\\bgi\\");
outtext("А это графический режим!");
getch();
closegraph();
```

```
cprintf("\nУвы, снова текстовый!");
getch();}
```

Существует 16 стандартных цветов (коды: 0-15). Функция setcolor(int color); меняет цвет на указанный в скобках, и дальше всё рисуется в этом цвете. Родственная ей функция setbkcolor(int color); меняет цвет фона. На цвет рисунка это не влияет. Какой установлен текущий цвет и цвет фона можно, узнать с помощью парных функций: getcolor(), getbkcolor(). В СИ под ДОС, цвет - целое число. В приложениях под Windows цвет - особый тип данных, состоящий из трёх компонент: красной, синей и зелёной. Там вы не связаны 16 цветами!

Простейшие фигуры в программировании называются графическими примитивами. Например, программа изображение линии, причём тип линии каждый раз меняется с помощью setlinestyle.

```
#include <graphics.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <conio.h>
//массив стилей линий
char *lname[] = {"SOLID_LINE", "DOTTED_LINE", "CENTER_LINE",
"DASHED_LINE", "USERBIT_LINE"};
int main(void)
{int gdriver = DETECT, gmode, errorcode;
int style, midx, midy;
char stylestr[40];
initgraph(&gdriver, &gmode, "c:\\tc\\bgi");
//проверка возможно ли инициализировать графику
errorcode = graphresult(); //возвращаем код ошибки
if (errorcode != grOk) //и если ошибка{
printf("Graphics error: %s\n", grapherrormsg(errorcode));
//Выводим стандартное сообщение
```

```
getch();
exit(1); //аварийный выход из программы}
midx = getmaxx() / 2; //находим координаты середины экрана
midy = getmaxy() / 2; //деля максимум по x и по y пополам
for (style=SOLID_LINE; style<=USERBIT_LINE; style++)
{setlinestyle(style, 1, 1); //устанавливаем стиль линии
strcpy(stylestr, lname[style]); //копируем в строку название типа
line(0, 0, midx-10, midy); //рисуем линию. Аргументы функции x1, y1, x2, y2.
rectangle(0, 0, getmaxx(), getmaxy()); //рисуем рамку
outtextxy(midx, midy, stylestr); //выводим по центру название стиля
getch();
cleardevice(); //очищаем экран}
closegraph(); //возвращаемся в текстовый режим
return 0;   }
```

Тип линии относится не только к команде line, но ко всем другим графическим примитивам. Рамки и всё остальное тоже будет пунктирным, если не вернуть стиль на исходный.

Ещё есть функция linerel(x,y), которая рисует линию из текущей позиции в позицию, указанную в аргументах и lineto(x,y), что сводит с данной точкой начало координат.

Задав координаты центра и радиус, мы нарисуем окружность:

circle(x, y, radius);

putpixel(x,y,color); выводит точку по координатам цвета color. Применив генератор случайных чисел, вы сможете вывести на экран звёздное небо.

Для рамки надо указать координаты левого верхнего и правого нижнего углов: bar(x1,y1,x2,y2);

3-мерная рамка применяется для вывода столбиковых диаграмм: bar3d(x1,y1,x2,y2, width, topflag); width - толщина рамки, число topflag задаёт наложение верха на рамку. Его обычно ставят 1.

bar3d(10,10,100,100,10,1);

Прямоугольник с координатами левого верхнего и правого нижнего углов выводят с помощью функции: rectangle(x1,y1,x2,y2);

Для изображения сектора Вы рисуете окружность, из которой вырезан сектор. Функцию удобно использовать для круговых диаграмм в экономических приложениях. Представьте, что центр окружности в начале координат. Начальный и конечный угол отсчитывается так, чтобы получился нужный сектор. Например, чтобы вырезать первую четверть окружности, начальный угол (stangle) будет =0, а конечный (endangle) = 90. Также надо указать и радиус.

```
#include<graphics.h>
#include<conio.h>
main() { int a=9, b=2;
initgraph(&a, &b,"");
setbkcolor(15); //цвет фона
setfillstyle(1,4); //задаём стиль заполнения сектора
pieslice(100,100,0,110,50); //сектор от 0 до 110 градусов
setfillstyle(1,14);
pieslice(100,100,110,160,50);
setfillstyle(1,1);
pieslice(100,100,160,280,50);
setfillstyle(1,2);
pieslice(100,100,280,360,50);  getch(); }
```

Для рисования многоугольника есть функция fillpoly, которая принимает массив парных координат точек и число точек, затем соединяет их линиями. Ваше дело - задать количество вершин своего многоугольника. Если их очень много, фигура получится «гладенькой» и можно нарисовать что-то хорошее. Раньше этот приём часто применялся в играх, если фигурка маленькая, то неровности заметно меньше.

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
int main(void)
{ int gdriver = DETECT, gmode, errorcode;
int i, maxx, maxy;
int poly[8];
initgraph(&gdriver, &gmode, "");
maxx = getmaxx(); //Получаем максимальные координаты по x и по y.
```

```
    maxy = getmaxy();
    poly[0] = 20; //1-я вершина
    poly[1] = maxy / 2;
    poly[2] = maxx - 20; //2-я вершина
    poly[3] = 20;
    poly[4] = maxx - 50; //3-я вершина
    poly[5] = maxy - 20;
    poly[6] = maxx / 2; //4-я - то же, что и первая
    poly[7] = maxy / 2;
    //Рисуем, меняя способы заполнения
    for (i=EMPTY_FILL; i<USER_FILL; i++){
    setfillstyle(i, getmaxcolor());
fillpoly(4, poly);
    getch(); }
    closegraph();
    return 0; }
```

Закрашивать можно эллипс (соответственно, и окружность), рамку (можно делать свои окна) и любую замкнутую фигуру, которую вы задали массивом точек в предудыщей программе.

Функция fillellipse(x,y, xradius, yradius); заполняет эллипс текущим стилем заполнения. У эллипса два радиуса по x и по y, поэтому надо указать оба. Если они равны, получится окружность. Нарисуем окно, как в Windows:

```
    #include <graphics.h>
    #include <conio.h>
    #include <dos.h>
    //Процедура вывода окна
void okno(int x1, int y1, int x2, int y2) {
    setfillstyle(1,7);
    bar(x1,y1,x2-1,y2-1);
    line(x1,y2-1,x2,y2-1);
    setcolor(0);
    line(x2-2,y2,x2-2,y1);                }
    //Рисуем кнопку
void button(int x1, int y1, int x2, int y2)      {
    setfillstyle(1,7);
    bar(x1,y1,x2-1,y2-1);
    line(x1,y2-1,x2-2,y2-1);
    setcolor(0);
    line(x2-2,y2-2,x2-2,y1);
    setcolor(15);
```

```
        line(x1,y1,x1,y2-2);
        setcolor(15);
        line(x1,y1,x2-2,y1);                    }
main()                                                  {
//Объявим все координаты
        int a=9,d=2;
        int x1=10,y1=10,x2=300,y2=100;
        int xb1=100,yb1=70,xb2=200,yb2=90;
        int xt=x1+35,yt=y1+10;
        int xc=xb1+40;
        int yc=yb1+6;
        initgraph(&a, &d, "c:\\tc\\bgi");
//Рисуем окно и кнопку
        okno(x1,y1,x2,y2);
        button(xb1, yb1, xb2, yb2);
//Выводим на них текст
        setcolor(0);
        outtextxy(xc,yc,"OK");
        outtextxy(xt,yt,"RII LTD presents...");
        setbkcolor(0);
        _AX=1;
        geninterrupt(0x33);
        getch();                                        }
```

Теперь посложнее - нарисуем пароходик, чтобы низ был красный, а верх - жёлтый.

```
        #include <graphics.h>
        #include <stdlib.h>
        #include <stdio.h>
        #include <conio.h>
int main(void)                  {
int gdriver = DETECT, gmode;
int i, maxx, maxy;
                        //Массивы координат: палубы,
int poly[18]={100,100,100,90,110,90,110,60,130,60,130,90,140,90,140,100,100,100};
                                                //корпуса
int poly1[8]={70,100,170,100,140,130,100,130};
          initgraph(&gdriver, &gmode, "");
        maxx = getmaxx();
        maxy = getmaxy();
                        //Рисуем: палубу,
        setfillstyle(1, 14);
        fillpoly(9, poly);
```

```
                        //корпус,
    setfillstyle(1, 4);
    fillpoly(4,poly1);
                        //дым,
    setfillstyle(1, 1);
    fillellipse(185,30,40,15);
    fillellipse(140,30,10,10);
    fillellipse(130,40,10,10);
    fillellipse(120,50,10,10);
                        //иллюминатор
    setfillstyle(1,0);
    fillellipse(119,80,8,8);
    outtextxy(170,25,"BIOS");
    getch();
    closegraph();
    return 0;                       }
```

Вывести текст можно с помощью функции outtext(char *text), где в скобках строка, которую надо вывести на экран. Но если вы хотите вывести строку в нужное место, используйте вывод из точки с заданными в функции координатами: outtextxy(int x, int y, char *text); Она позволяет выводить текст так же легко, как в текстовом режиме при использовании gotoxy(x,y)!

Русский шрифт поменять можно, только изменив разрешение экрана, тогда он будет большой и некрасивый. Иначе можно поступать только с латиницей. Шрифты хранятся в папке BGI и носят расширения **.chr. Задать размер текста, а также его шрифт поможет функция settextstyle();

```
    #include <graphics.h>
    #include <stdlib.h>
    #include <stdio.h>
    #include <conio.h>
                                //массив шрифтов
    char *fname[] = { "DEFAULT font","TRIPLEX font","SMALL font",
    "SANS SERIF font","GOTHIC font"};
int main(void)                  {
    int gdriver = 9, gmode=0;
    int style, midx, midy;
    int size = 1; //размер шрифта
    initgraph(&gdriver, &gmode, ""); //Драйвер в текущей папке - пишем кавычки
    midx = getmaxx() / 2; //вычисляем середину экрана
    midy = getmaxy() / 2;
//задаём выравнивание по центру по x и по y
```

```
        settextjustify(CENTER_TEXT, CENTER_TEXT);
//прокручиваем все шрифты и выводим на экран
        for (style=DEFAULT_FONT; style<=GOTHIC_FONT; style++) {
        cleardevice();
        settextstyle(style, HORIZ_DIR, size); // - вывод текста слева-направо
//пишем по центру названия из массива fname
        outtextxy(midx, midy, fname[style]);
//увеличиваем размер шрифта на 1
        size++;
getch();                                                        }
        closegraph();
        return 0;                       }
```

Кроме вывода слева-направо, можно задать и другое расположение шрифта: VERT_DIR - вертикальное расположение.

С помощью функций рисования графических примитивов и функций копирования изображения (putimage/getimage) на экран можно выводить фигурки типа колобков и яблок, которыми пользователь сможет управлять с клавиатуры. Сделаем заготовку для игры, где по экрану перемещается существо под управлением с клавиатуры. Применив знания по массивам и рисованию графических примитивов, вы без труда нарисуете ему лабиринт, в котором он будет существовать. Будем рисовать колобка. Нарисуем его с помощью сектора pieslice - кусок пирога.

```
        #include <graphics.h>
        #include <stdlib.h>
        #include <stdio.h>
        #include <conio.h>
        //Прототип функции рисования. Сама функция будет объявлена позднее
void draw(int x, int y);
int main(void)                                  {
        int gdriver = DETECT, gmode;
        void *ptr; //резервируем пустой указатель - стартовая ячейка памяти,
                        // начиная с которой мы будем хранить наше изображение
          int x, y, maxx;
          unsigned int size;
        initgraph(&gdriver, &gmode, "");
        maxx = getmaxx();
//Начальные координаты
        x = 30;
        y = 30;
```

```
//Цвет фона
      setbkcolor(2);
//Рисуем героя
      draw(x, y);
//Вычисляем, сколько изображение будет занимать в памяти
       size = imagesize(x-10, y-10, x+10, y+10);
//Выделяем память начиная с того места, куда указывает указатель
      ptr = malloc(size);
//Копируем в память изображение
      getimage(x-10, y-10, x+10, y+10, ptr);
       int key;
       int nx=x;
       int ny=y;
       do
          {key=getch(); //Обработка нажатия клавиш
           switch(key){
           case 80: ny+=10; break;
          case 72: ny-=10; break;
          case 77: nx+=10; break;
          case 75: nx-=10; break;}
//Стираем старое изображение наложением на него нового
      putimage(x-10, y-10, ptr, XOR_PUT);
//Меняем координаты соответственно нажатой клавише
      x = nx;
      y=ny;
//Выводим изображение
      putimage(x-10, y-10, ptr, XOR_PUT);
         }while(key!=27);
//Освобождаем память
      free(ptr);
//Закрываем графику
      closegraph();
      return 0;                                          }
//Функция рисования. Сюда можно запрограммировать любой рисунок
void draw(int x, int y)
      {setcolor(14);
      setfillstyle(1,14);
      moveto(x, y);
      pieslice(x,y,20,350,10);
      setfillstyle(1,4);
      fillellipse(x+1,y-4,2,2);}
```

На досуге подумайте, как из этой заготовки сделать полноценную игру!

С помощью функции getimage и putimage изобразить фигуру, движущуюся случайным образом по экрану на фоне звездного неба.

Примечание: Звездным небом называется множество точек на экране, цвет и координаты которых вычисляются с помощью генератора случайных чисел. Код на C++ :

```
#include <graphics.h>
#include <conio.h>
#include <stdlib.h>
#include <dos.h>
int change(int current)                    {
   if (random(32767) > 30000)
   {        switch (random(5))
      {   case 0: return -2;
          case 1: return -1;
          case 2: return 0;
          case 3: return 1;
          case 4: return 2;
       }
   }
   return current;                          }
void main(void)
{   int gd = VGA, gm = VGAHI, i;
   int StartX = 100, StartY = 50, r = 20;
   int ulx, uly, lrx, lry, width, height, size;
   int dx = 0, dy = 0;
   int do_job = 1;
   void * img;
  initgraph(&gd, &gm, "c:\\temp\\bc31\\bgi");
  setfillstyle(SOLID_FILL, getmaxcolor());
   fillellipse(StartX, StartY, r, (r / 3) + 2);
   ellipse(StartX, StartY - 4, 190, 357, r, r / 3);
   line(StartX + 7, StartY - 6, StartX + 10, StartY - 12);
   circle(StartX - 10, StartY - 12, 2);
   line(StartX - 7, StartY - 6, StartX - 10, StartY - 12);
   circle(StartX + 10, StartY - 12, 2);
   ulx = StartX - (r + 1);
   uly = StartY - 14;
   lrx = StartX + (r + 1);
   lry = StartY + (r / 3) + 3;
   width = lrx - ulx + 1;
   height = lry - uly + 1;
   size = imagesize(ulx, uly, lrx, lry);
   img = malloc(size);
   getimage(ulx, uly, lrx, lry, img);
   cleardevice();
   for (i = 1; i < 1000; i++)
      putpixel(random(640), random(480), random(16));
   while (do_job)
   { putimage(StartX, StartY, img, XOR_PUT);
```

```
    delay(50);
    putimage(StartX, StartY, img, XOR_PUT);
    dx = change(dx);
    dy = change(dy);
    StartX += dx;
    StartY += dy;
    if (StartX > 639 - width) StartX = 639 - width;
    if (StartY > 479 - height) StartY = 479 - height;
    if (StartX < 0) StartX = 0;
    if (StartY < 0) StartY = 0;
    if (kbhit()) do_job = (getch() != ' ');  }
  closegraph();  }
```

Нарисовать замкнутый многоугольник и заполнить его буквами размером 8x8 пикселов. Буква "B".

```
#include <graphics.h>
#include <conio.h>
main()
{unsigned char pattern[]={56,36,36,60,34,34,34,62};
 int poly[] ={10,10,15,20,50,50,100,25,90,5};
 int graphdriver=DETECT, graphmode;
 initgraph(&graphdriver, &graphmode, "");
 setfillpattern(pattern, WHITE);
 fillpoly(5, poly);
 getch(); closegraph();
}
```

СПИСОК ЛИТЕРАТУРЫ

1. Архангельский А.Я. Язык C++ в C++Builder: Справ. и метод. пособие [текст]/ А.Я. Архангельский.- М: БИНОМ, 2008. -944 с.
2. Герберт Шилдт. C: полное руководство, классическое издание = C: The Complete Reference, 4th Edition. М.: Вильямс, 2010. -704с.
3. Гукин Д. Язык программирования Си для «чайников» = C For Dummies. - М.: Диалектика, 2006.-352с.
4. Искусство программирования, том 1. Основные алгоритмы, 3-е издание Дональд Э. Кнут.
5. Искусство программирования, том 4 А. Комбинаторные алгоритмы, часть 1, , часть 1, Дональд Эрвин Кнут.
6. Искусство тестирования программ, 3-е издание, Гленфорд Майерс, Том Баджетт, Кори Сандлер.
7. Камаев В.А. Технологии программирования: Учебник/ В.А. Камаев, В.В. Кастерин. -2-е изд., перераб. и доп. –М.: Высш. шк., 2006. -454с.
8. Кармайкл Э., Хейвуд Д. Быстрая и качественная разработка программного обеспечения: Пер. с англ. –М.: Издательский дом «Вильямс», 2003. -400с.
9. Керниган Б., Ритчи Д. Язык программирования Си = The C programming language.— 2-е изд. М.: Вильямс, 2007.-304с.
10. Керниган Б., Ричи Д. Язык программирования СИ. – М.: Мир, 1985.
11. Кочан С. Программирование на языке Си = Programming in C.— 3-е изд. М.: Вильямс, 2006.-496с.
12. Освой самостоятельно C за 21 день, 6-е издание Брэдли Джонс, Питер Эйткен; Вильямс, 2005. -800с.
13. Поган А.М., Царенко Ю.А. Программирование в С. Просто как дважды два / А.М. Поган, Ю.А. Царенко. –М.: Эксмо, 2006. – 320с.
14. Прата С. Язык программирования С: Лекции и упражнения C Primer Plus. М.: Вильямс, 2006. -960с.
15. Самюэл П. Язык программирования С: [текст]: Пер. с англ/ П. Самюэл, Л. Гай. -М.: Бином-Пресс, 2009. -528 с.

16. Сахань Ю.В. Задания для курсовых работ по алгоритмическим языкам. Методические указания и задания по выполнению курсовых работ для студентов специальности 073000 «Прикладная математика» Рубцовский индустриальный институт.- Рубцовск, 2006.– 34с.
17. Сахань Ю.В. Языки программирования. Часть 2 – СИ: Методические указания и задания к лабораторным работам для студентов специальности 073000 «Прикладная математика» Рубцовский индустриальный институт.- Рубцовск, 2006.– 41 с.
18. Справочник программиста по C/C++, 3-е издание Герберт Шилдт; Вильямс, 2006. -432с.
19. Эпштейн М.С. Практикум по программированию по С: [текст]: Учеб. пособие для ссузов/ М.С. Эпштейн. -М.: Академия, 2007. -128 с.
20. Язык программирования C (Си), 2-е издание Брайан У. Керниган, Деннис М. Ритчи; Вильямс, 2012. -304с.
21. Язык программирования C (Си). Лекции и упражнения, 5-е издание Стивен Прата; Вильямс, 2012. -960с.
22. Язык программирования Си для "чайников", 2-е издание Дэн Гукин; Диалектика, 2006. -352с.

ПРИЛОЖЕНИЕ 1

Расширенная кодовая таблица

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

ПРИЛОЖЕНИЕ А

Примеры тестов текущего контроля знаний

Текущий контроль знаний проводится дважды в семестр на аттестационных неделях в виде письменного опроса по следующим тестам:

Тест 1 – Типы данных. Константы. Операции.

- Перечислите основные разделы программы и приведите пример описания каждого раздела.
- Перечислите лексемы и приведите пример для каждого вида.
- Определите тип констант:
    - *0X2FL*
    - *2.1E5*
    - *053*
    - *12345*
    - *‘x’*
    - *0XFFFL*
    - *345.*
    - *"Строка"*
- Установите соответствие:
-

float	*2 байта*
short int	*1 байт*
unsigned	*10 байт*
long float	*4 байта*
double	*4 байта*
long double	*4 байта*
char	*2 байта*
long int	*8 байт*

- Назовите виды операций и приведите примеры для каждого вида.

- Укажите приоритет выполнения операций:
    - *&& – – ! < = = * % != || + + >=*
    - *a = - (b * = c + a * d % 2) / 3 && ! ++ k;*
    - *x - = k = = x || ! y * ++ a + ~ c << b % 3;*
    - *x = (a != b <= ! k && c > - k, c || a & sizeof(-1));*
    - *k =! j != !m;*
- Вычислите выражение:
    - *7&5*
    - *3|8*
    - *7|8*
    - *5|3*
    - *7>>2*
    - *15<<3*
    - *8<<4*
    - *116>>4*
- Запишите выражение:
    - Дано целое число ***x***. Если оно положительное, то увеличить его на 1, иначе уменьшить на 1.
    - Определить знак числа x.
    - Является ли введенное целое число однозначным.
    - Является ли введенное целое число двузначным.

Тест 2 – Типы данных. Операции. Ввод-вывод.

- Назовите атрибуты объекта.
- Перечислите классы хранения объектов.
- Запишите структуру перечислимого типа данных.
- Сформулируйте правила использования перечислений.
- Перечислите операции языка СИ. Укажите приоритет операций.
- Что получится в результате выполнения операций?

```
  00110101
& 11000101
```

- ……..

```
  00110101
| 11000101
```

- ……..

```
  00110101
^ 11000101
```

- ……..

- Что получится в результате выполнения операции сдвига? Представьте результат в 10-м, 2-м и 16-м виде.
    - 01110100>>3
    - 01110100<<3
- Какие из функций ввода неправильные и почему?
    - scanf(“%d”, &a);
    - scanf(“%f %f”, &b, c);
    - scanf(“Введите число”, &k);
    - scanf(“%d”, a2);
    - scanf(“%d”, a1[i]);
    - scanf(&d);
- Какие из функций вывода неправильные и почему?
    - printf(“%d”, &a);
    - printf(“Y=”, y);
    - printf(“%5.3d”, c);
    - printf(“%d”, d);
    - printf(“%7.2f”, p);
    - printf(“%cc”, c1, c2);

Тест 3 – Массивы. Строки.

- Определение вектора.

- Ответьте на следующие вопросы:
    - *Может ли массив содержать один элемент? А ни одного?*
    - *Можно ли во время выполнения программы изменить размер массива?*
    - *Может ли типом индекса быть тип integer или real?*
- Физическая и логическая структура вектора.
- Опишите ввод и вывод элементов одномерного массива.
- Какие из приведенных описаний массива неправильные и почему:
    - *int b[25]*
    - *char c[2,5];*
    - *float d:array[char] of 2..10;*
    - *myt e[-3];*
    - *f:array[–5..5] of integer;*
- Каким идентификатором определяются данные строкового типа? Приведите пример.
- Какова максимально возможная длина строки?
- Как обратиться к отдельным символам строки? Приведите пример.
- Перечислите основные функции для работы со строками.
- Запишите различные варианты описания двумерного массива.
- Как на экране будет выглядеть двумерный массив, если вывод его элементов организовать следующим образом:
    - *for (i=0; i<=n; i++) {*
      for (j=0; j<=m; j++) printf("%d ",a[i, j]); printf("\n"); }
    - *for (i=0; i<=n; i++) {*
      for (j=0; j<=m; j++) printf("%d ",a[i, j]);}
    - *for (i=0; i<=n; i++) {*
      for (j=0; j<=m; j++) printf("a[%d, %d]=%d", i, j, a[i, j]); }
- Какие из указанных ниже операторов присваивания неправильны?
    - *a:=b;*

- *a[1]=a[8];*
- *b[3]=a[3];*
- *b[1]=b[8];*
- *x=a[5];*
- *b[5]=x;*

- Опишите два множества *R* и *L*, содержащие русские и латинские буквы соответственно.

Тест 4 – Записи.

- Дайте определение записи.
- Опишите структуру, состоящую из полей:
  - *фамилия, год рождения, адрес*
  - *код товара, наименование, цена*
  - *фамилия, группа, оценки за сессию*
  - *название месяца, температура за каждый день, среднемесячная температура*
- Определите объем памяти, требуемый для размещения каждой записи из предыдущего пункта.
- Дайте определение объединения.
- Чем отличается объединение от структур?

Тест 5 – Файлы.

- Назовите отличительные черты текстовых и двоичных файлов.
- Ответьте на вопросы:
- *Верно ли, что элементы файла должны быть одного типа и что файл отличается от массива только тем, что размер файла произволен, а размер массива фиксирован?*
- *Можно ли, считая из файла пятый элемент, затем сразу же считать второй элемент?*
- *В какое место файла можно добавлять новые элементы: в начало, в середину, в конец, куда угодно, никуда?*

ПРИЛОЖЕНИЕ Б

Примерные вопросы билетов итогового контроля знаний

1. История языка Си и его особонности.
2. Какова структура программы на языке Си? Объясните каждую составляющую.
3. Перечислите константы языка и приведите пример для каждого вида.
4. Работа с отладчиком.
5. Назовите классификацию операций, используемых в языке Си, и приведите примеры для каждого вида. Укажите приоритет выполнения операций.
6. Назовите атрибуты объекта. Перечислите классы хранения объектов.
7. Запишите определение данного перечислимого типа. Объясните понятие этого типа данных. Приведите пример использования перечислений.
8. Операторы присваивания и условные.
9. Операторы цикла в Си.
10. Вспомагательные управляющие операторы.
11. Классы памяти. Массивы.
12. Определение вектора.
13. Физическая и логическая структура вектора.
14. Опишите ввод и вывод элементов одномерного массива.
15. Как определяются данные строкового типа? Приведите пример. Какова максимально возможная длина строки?
16. Как обратиться к отдельным символам строки? Приведите пример.
17. Перечислите основные стандартные функции для работы со строками.
18. Запишите различные варианты описания двумерного массива.
19. Способы внутренней сортировки данных.
20. Функции ввода и вывода и их классификация.

21. Функции ввода – вывода нижнего уровня.
22. Функции ввода – вывода потока.
23. Функции ввода – вывода для консоли и порта.
24. Дайте определение и опишите применение записи.
25. Опишите структуру, состоящую из полей: *фамилия, группа, оценки за сессию.* Определите объем памяти, требуемый для размещения структуры.
26. Дайте определение объединения (смеси). Отличие объединения от структуры.
27. Понятие «запись» в Си, описание и применение в программах.
28. Макроопределения и препроцессор.
29. Функции внутренние, внешние и стандартные.
30. Организация проекта в Си.
31. Назовите отличительные черты текстовых и двоичных файлов.
32. Стандартные функции для работы с файлами.
33. Динамические переменные и объекты.
34. Функции для работы с указателями (адресами).
35. Условная трансляция. Прагмы.
36. Инсталляция графического режима.
37. Установка цветов и стилей в графике.
38. Функции включения точек и вычерчивания линий.
39. Изображение графических примитивов: прямоугольник, эллипс, многоугольник.
40. Организация анимации в графике.
41. Использование текста и окон в графике.
42. Прямой доступ к видеопамяти.

ПРИЛОЖЕНИЕ В

Примеры тестов контроля остаточных знаний

Вариант 1

- Комментарии в программе обозначаются:
    - /*…*/
    - {…}
    - //
    - {*…*}
- Лексемами могут быть:
    - Идентификаторы
    - Переменные
    - Ключевые слова
    - Выражения
- Символьные константы могут быть представлены:
    - Печатными символами
    - Управляющими кодами
    - Непечатными символами
- Пример описания константы с плавающей точкой:
    - 345.
    - 3.14159
    - 2.1E5
    - .123E3403e-5
- Инструкции для обработки информации могут быть следующими:
    - выполняющимися только тогда, когда верны специфические условия
    - линейными
    - повторяющимися несколько раз
    - циклическими
- Атрибуты у объекта в Си следующие:
    - область действия
    - имя
    - класс хранения
    - время жизни
    - тип

- Тернарные операции:
    - *, /, %, +, –
    - &, |, ^, <<, >>
    - выражение_1 ? выражение_2 : выражение_3
    - &&, ||
    - =
    - операнд_1 ? операнд _2 : операнд _3
    - –, ~, !, *, &, + +, –, sizeof()
    - <, >, <=, >=, ==, !=
- К вспомогательным операторам относятся:
    - BREAK
    - IF (выражение) оператор_1; [ELSE оператор_1;]
    - CONTINUE
    - DO оператор; WHILE (выражение);
    - RETURN(выражение);
    - *GOTO*
- Описание структуры имеет следующий формат:
    - *struct {список элементов и их типов} [идентиф.1, идентиф.2, ...];*
    - *спецификатор типа [идентификатор] : константное выражение;*
    - *struct [ярлык]{список элементов и их типов} [идентиф.1, идентиф.2, ...];*
    - *struct [ярлык] [идентиф.1, идентиф.2, ...];*
- Назовите функции ввода-вывода для консоли
    - *access, chmod, chsize, close, creat, creat, open, sopen, read, write, setmode*
    - *inp, inpw, outp, outpw*
    - *fopen, fdopen, freopen, fread, putchar, fwrite, fscanf, sprintf, ftell, fsetpos, rewind, fclose*
    - *setbuf, setvbuf*
    - *access, chmod, chsize, close, creat, creat, open, sopen, read, write, setmode*
    - *stdin, stdout, stderr*
    - *cgets, cprintf, cputs, getch, cscanf, getche, kbhit, putch, ungetch*

Вариант 2

- Включение внешних файлов осуществляется с помощью директивы:
    - #INCLUDE <имя файла>

    - DEFINE “путь”
    - #INCLUDE “путь”
    - #DEFINE <имя> <значение>
- Лексемами могут быть:
    - Строки
    - Функции
    - Операции
    - Разделитель
- Константы могут быть следующих типов:
    - Целые
    - Плавающие
    - Перечислимые
    - Символьные
- Пример описания шестнадцатеричной константы:
    - 15236
    - 00
    - 052L
    - –689
    - 0XA015
    - 0X2FL
- Инструкции для обработки информации могут быть следующими:
    - повторяющимися несколько раз
    - циклическими
    - выполняющимися в различных местах программы
    - условными
- В языке СИ предусмотрены классы хранения объектов:
    - внутренний автоматический
    - регистровый
    - внутренний статический
    - внутренний
- Бинарные операции:
    - *, /, %, +, –
    - &, |, ^, <<, >>

- выражение_1 ? выражение_2 : выражение_3
- &&, ||
- =
- операнд_1 ? операнд _2 : операнд _3
- –, ~, !, *, &, + +, –, sizeof()
- **<, >, <=, >=, ==, !=**

- К операторам цикла относятся:
  - RETURN
  - SWITCH (выражение) { [CASE <константное выражение> : <список операторов>;]…[DEFAULT : <список операторов>;] }
  - WHILE (выражение) оператор;
  - FOR ([выражение инициализации];[условное выражение]; [выражение итерации]) оператор;
  - CONTINUE
  - DO оператор; WHILE (выражение);
  - GOTO
- Описание объединения имеет следующий формат:
  - *union [ярлык]{список элементов и их типов} [идентиф.1, идентиф.2, ...];*
  - *union спецификатор типа [идентификатор] : константное выражение;*
  - *union [ярлык] [идентиф.1, идентиф.2, ...];*
  - *спецификатор типа [идентификатор] : константное выражение;*
- Назовите функции ввода-вывода для порта
  - *access, chmod, chsize, close, creat, creat, open, sopen, read, write, setmode*
  - *inp, inpw, outp, outpw*
  - *fopen, fdopen, freopen, fread, putchar, fwrite, fscanf, sprintf, ftell, fsetpos, rewind, fclose*
  - *setbuf, setvbuf*
  - *access, chmod, chsize, close, creat, creat, open, sopen, read, write, setmode*
  - *stdin, stdout, stderr*
  - *cgets, cprintf, cputs, getch, cscanf, getche, kbhit, putch, ungetch*

Вариант 3

- Головная функция описывается:
  - INTEGER MAIN()

    - VOID MAIN( )
    - MEIN()
    - MAIN(VOID)
- Лексемами могут быть:
    - Ключевые слова
    - Выражения
    - Строки
    - Функции
- Целые константы делятся на:
    - Десятичные
    - Восьмеричные
    - Двоичные
    - Шестнадцатеричные
- Пример описания восьмеричной константы:
    - 15236
    - 00
    - 052L
    - –689
    - 0XA015
    - 0X2FL
- Назовите элементы программирования:
    - Переменные
    - Условное выполнение
    - Циклы
    - Функции
    - Константы
- В языке СИ предусмотрены основные типы данных:
    - char
    - unsigned
    - int
    - short
    - float
    - long

- Операции языка Си делятся на:
    - Унарные
    - Отношения
    - Бинарные
    - Арифметические
    - Тернарные
    - Логические
- К условным операторам относятся:
    - *RETURN*
    - *SWITCH (выражение) { [CASE <константное выражение> : <список операторов>;] …[DEFAULT : <список операторов>;] }*
    - *;*
    - *BREAK*
    - *IF (выражение) оператор_1; [ELSE оператор_1;]*
    - *CONTINUE*
- Описание указателя имеет следующий формат:
    - *[тип данных] *<идентификатор>;*
    - *<тип данных> *<идентификатор>;*
    - *<тип данных> *[идентификатор];*
- Назовите функции ввода-вывода верхнего уровня
    - *access, chmod, chsize, close, creat, creat, open, sopen, read, write, setmode*
    - *inp, inpw, outp, outpw*
    - *fopen, fdopen, freopen, fread, putchar, fwrite, fscanf, sprintf, ftell, fsetpos, rewind, fclose*
    - *setbuf, setvbuf*
    - *access, chmod, chsize, close, creat, creat, open, sopen, read, write, setmode*
    - *stdin, stdout, stderr*
    - *cgets, cprintf, cputs, getch, cscanf, getche, kbhit, putch, ungetch*

Вариант 4

- Головная функция описывается:
    - MAIN(INTEGER)
    - VOID MAINE( )
    - MAIN()

    - MAIN(VOID)
- Лексемами могут быть:
    - Выражения
    - Константы
    - Операторы
    - Разделитель
- Константы могут быть следующих типов:
    - Перечислимые
    - Символьные
    - Строковые
    - Логические
- Пример описания десятичной константы:
    - 15236
    - 00
    - 052L
    - –689
    - 0XA015
    - 0X2FL
- Назовите элементы программирования:
    - Ввод
    - Процедуры
    - Типы данных
    - Операции
    - Вывод
- В языке Си предусмотрены классы хранения объектов:
    - внешний
    - внешний статический
    - автоматический
    - статический
- Унарные операции:
    - *, /, %, +, –
    - &, |, ^, <<, >>
    - выражение_1 ? выражение_2 : выражение_3

- &&, ||
- =
- операнд_1 ? операнд _2 : операнд _3
- –, ~, !, *, &, + +, –, sizeof()
- <, >, <=, >=, ==, !=

- К вспомогательным операторам относятся:
  - *RETURN*
  - *SWITCH (выражение) { [CASE <константное выражение> : <список операторов>;] ...[DEFAULT : <список операторов>;] }*
  - *WHILE (выражение) оператор;*
  - *FOR ([выражение инициализации];[условное выражение]; [выражение итерации]) оператор;*
  - *;*
- Описание массива имеет следующий формат:
  - *<спецификатор типа> <описатель> [константное выражение];*
  - *<спецификатор типа> [описатель] [];*
  - *<спецификатор типа> <описатель> [константное выражение];*
  - *<спецификатор типа> [описатель] [];*
- Назовите функции ввода-вывода нижнего уровня
  - *access, chmod, chsize, close, creat, creat, open, sopen, read, write, setmode*
  - *inp, inpw, outp, outpw*
  - *fopen, fdopen, freopen, fread, putchar, fwrite, fscanf, sprintf, ftell, fsetpos, rewind, fclose*
  - *setbuf, setvbuf*
  - *access, chmod, chsize, close, creat, creat, open, sopen, read, write, setmode*
  - *stdin, stdout, stderr*
  - *cgets, cprintf, cputs, getch, cscanf, getche, kbhit, putch, ungetch*

ПРИЛОЖЕНИЕ Г

Задания для практических занятий

Практическое занятие 1. Структура программы Си. (2ч)

- Вычислить модуль вещественного числа ***x*.**
- Определить знак числа x.
- Вычислить значение выражения:
- $y = 4x - 8x^2$
- $y = \frac{2x^2 - 5}{x^4 + 2}$
- $y = \frac{\sin x}{2x^2 + 3}$
- Дано целое число ***x***. Если оно положительное, то увеличить его на 1, иначе уменьшить на 1.
- Проверить на четность данное целое число.
- Даны два числа, ***n***, ***k***. Наибольшее из них заменить нулем.
- Проверить, равно ли наименьшее из двух чисел, ***n***, ***k,*** нулю.
- Даны два числа, ***n***, ***k***. Наименьшее из них заменить единицей.
- Дано целое число ***x***. Если оно четное, то увеличить его в 2 раза, иначе оставить без изменения.
- Даны два числа, ***n***, ***k***. Вывести значение максимального из этих чисел.
- Является ли введенное целое число однозначным?
- Является ли введенное целое число двузначным? !(a/10)&a/10? "двузначное" : "не двузначное".
- Даны два числа, ***n***, ***k***. Они оба должны принять значения наибольшего из этих чисел.

Практическое занятие 2. Операции языка Си. (2ч)

- Установить порядок выполнения операций:
- a=-(b*=c+a*d%2)/3&&!++k;
- x-=k==x||!y*++a+~c<<b%3;

- x=(a!=b<=!k&&c>-k,c||a&sizeof(-1));
- k=!j!=!m;
- Установить все битовые переключатели натурального числа в состояние выключить (включить).
- Вычислить выражения:
- 134&28;
- 134|28;
- 134^28;
- 215|68;
- 215&68;
- 215^68;
- 111&321;
- 111|321;
- 111^321;
- Какую маску нужно задать, чтобы проанализировать (включить, выключить) 0, 3, 7биты (&, |). Записать маску в десятичном и шестнадцатеричном виде.
- Организовать сдвиг влево для числа x=7 на 1, 2, 3 разряда. Результат записать в двоичном десятичном, шестнадцатеричном виде.
- Организовать сдвиг вправо для числа x=125 на 1, 2, 3 разряда. Результат записать в двоичном десятичном, шестнадцатеричном виде.
- Составить программу преобразования строчных букв в заглавные и, наоборот, заглавные в строчные, используя поразрядные операции.
- Составить программу вывода битового представления беззнакового числа, т.е. значение каждого бита.
- с использованием поразрядных операций;
- без использования поразрядных операций.

Практическое занятие 3. Основные операторы. (3ч)

- Даны три числа. Если ни одно не равно нулю, то найти произведение

чисел, иначе сумму.

- Вывести на терминал число, получаемое выписыванием в обратном порядке цифр заданного натурального числа (сформировать число).
- Дана последовательность из ста целых чисел. Определить количество чисел в наиболее длинной последовательности из подряд идущих нулей.
- Составить программу распечатки всех пар целых чисел x и ***y***, которые являются решением уравнения $2y-x^2=4$, $x[-50;25]$, $y[100;300]$.
- Составить программу, в которой определяется, попадает ли заданная точка с координатами ***(x,y)*** в область ***L***.

- Дано целое число. Определить, сколько в нем цифр.
- Квадрат любого натурального числа равен сумме n первых нечетных чисел. Найти квадрат числа, используя указанную закономерность.

$1^2=1$

$2^2=1+3$

$3^2=1+3+5$

$4^2=1+3+5+7$ …

- Найти куб числа, используя указанную закономерность.

$1^3=1$

$2^3=3+5$

$3^3=7+9+11$

$4^3=13+15+17+19$

- Вывести на экран самую большую цифру числа.
- Треугольник, у которого длины сторон и площадь представляют собой целые числа, называется треугольником Герона. Например, 13, 14, 15, 84. Определить, является ли данный треугольник со сторонами ***a, b, c*** треугольником Герона.

Практическое занятие 4. Одномерные массивы. (2ч)

- Создать массив, заполнить его числами Фибоначчи: 1 1 2 3 5 8 13 21 …
- Вывести на терминал все двузначные элементы одномерного массива.
- Дан одномерный массив. Определить, сколько в нем пар одинаковых соседних элементов.
- Дан массив. Переписать его элементы в обратном порядке.
- Вывести на экран те элементы массива, индексы которых являются степенями двойки.
- Даны два массива. Найти наименьший элемент первого массива, который не входит во второй массив (считая, что хотя бы один такой элемент есть).
- Написать программу, которая определяет количество ненулевых элементов массива.
- Написать программу, которая проверяет, являются ли элементы массива возрастающей последовательностью.

Практическое занятие 5. Строки, матрицы. (2ч)

- Написать программу, которая удаляет из введенной с клавиатуры строки начальные пробелы.
- Даны две строки. Удалить из первой строки символы, которые входят во вторую строку.
- Создать матрицу [5][5], значение каждого элемента равно сумме номера строки и номера столбца, на пересечении которых он находится.
- Найти сумму минимальных элементов главной и побочной диагонали квадратной матрицы.
- Поменять местами минимальный и максимальный элементы матрицы.
- Дан одномерный массив A[16]. Переписать все данные из него в массив B[5][5].
- Определить, является ли матрица симметричной относительно главной диагонали.
- Найти минимальный элемент матрицы, поменять местами столбцы:

содержащий этот элемент и первый.

Практическое занятие 6. Структуры, объединения. (2ч)

- Дано описание:

struct a1{char fam[16];

int i;

union a2{char c;

int k;}a;

struct a3{unsigned f1:6;

unsigned f2:4;

unsigned :0;

unsigned f3:8;} b; } c;

- Как обратиться к полю структуры *a1*?
- Как обратиться к полю структуры *a2*?
- Как обратиться к полю структуры *a3*?
- Опишите битовую структуру, состоящую из полей.
- Текущая дата: число, месяц, год.
- Текущее время: часы, минуты, секунды.
- Данные о студентах: курс, номер группы, оценка, рейтинг.
- Написать программу, которая обрабатывает результат экзамена. Для каждой оценки программа должна вычислять процент от общего количества оценок.
- Написать программу, которая подводит итоги олимпийских игр. Вводится название страны и число медалей разного достоинства. Вычислить общее количество медалей и соответствующее ему число очков. После этого упорядочить список в соответствии с набранным количеством очков.
- Золото – 7 очков
- Серебро – 6 очков
- Бронза – 5 очков
- Данные об акционерах некоторого предприятия представлены в виде

структуры: Фамилия, И., О., Список акций (количество и стоимость каждого вида)

- Создать массив от структуры.
- Вывести все данные.
- Вывести держателей акций определенной стоимости.
- Вывести список акционеров, имеющих наибольшее количество акций.

Практическое занятие 7. Работа с файлами. (2ч)

- Дан файл целых чисел. Подсчитать количество цифр в файле. Подсчет оформить функцией.
- Создать файл целых чисел. Определить, сколько раз в нем встречается введенное с клавиатуры число.
- Дан текстовый файл, состоящий из строк. Заполнить другой файл, состоящий из строк первого, дополненных строчками данных о самой длинной строке исходного файла.
- Создать файл со структурой:
  - Наименование товара
  - Цена товара
  - Количество
  - Фамилия продавца

По фамилии продавца вывести сумму проданных товаров.

- Создать файл целых чисел. Вывести на экран все числа взаимно простые суммой своих цифр (Ненулевые целые числа n и m называются **взаимно простыми**, если HOD(n, m) = 1.). Найти максимальное простое 3-значное число.
- Создайте файл целых чисел. Напишите программу, переписывающие компоненты файла в обратном порядке.
- Имеется текстовый файл. Составить программу, которая, игнорируя исходное деление файла на строки, переформатирует его, разбивая на строки так, чтобы каждая строка оканчивалась точкой либо содержала ровно 60 литер, если среди них нет точки.
- Файл содержит последовательность слов, содержащих от 1 до 9 букв (слова

разделены пробелом). Вывести на экран все слова наименьшей длины.

- Дан файл действительных чисел. Описать функцию ***incr(f)***, определяющую количество элементов в наиболее длинной возрастающей последовательности файла ***f***.

- Описать функцию lines(f), которая построчно выводит содержимое непустого текстового файла f, вставляя в начало каждой строки ее порядковый номер (4 символа) и пробел.

Практическое занятие 8. Динамические структуры, функции. (3ч)

- Дано четное число N>2. Проверить для этого числа гипотезу Гольбаха: каждое четное число >2 можно представить в виде суммы двух простых чисел. Проверить, является ли число простым, с помощью функции.

- Написать функцию нахождения наибольшего общего делителя двух чисел. Ввод и вывод чисел в головной функции.

- Описать функцию, которая подсчитывает количество пустых строк в текстовом файле.

- Дан файл картотеки сведений об автомобилях (марка, номер, фамилия владельца).

- Создать списковую структуру.

- Найти данные об автомобиле по фамилии владельца.

- Исключить из картотеки сведения об автомобиле по указанному номеру.

- Создать очередь (односвязный список) из заранее неизвестного количества целых чисел, введенных с клавиатуры. Простые числа переписать в двусвязный список, удалив из односвязного.

- Описать функцию, которая подсчитывает число вхождений элемента L в список.

- Описать функцию, которая удаляет из непустого списка все вхождения элемента L.

- Написать программу, которая переводит выражение, записанное в обычной (инфиксной) форме в текстовом файле, в постфиксную форму и записывает это

выражение в другой файл.

Задачи

- Напишите расширенную версию библиотеки функций для работы с комплексными числами. Реализуйте операции сложения, вычитания, умножения и деления, а также функции, вычисляющие аргумент и модуль комплексного числа. Реализуйте функции комплексного аргумента sinc, cosc, expc, logc, asinc , acosc. Чему равен арккосинус двух (acosc(2) = ?) ? Напишите программу, тестирующую вашу библиотеку.

- Напишите библиотеку функций по теме "аналитическая геометрия"
  - Решать задачи на плоскости:
    - вычислять расстояние для пар (точка, точка), (точка, прямая),
    - находить площадь многоугольника,
    - проверять, пересекаются ли две прямые, и находить точку их пересечения,
    - проверять, пересекаются ли два отрезка, и находить точку их пересечения.
  - Решать задачи в пространстве:
    - вычислять расстояние для пар (точка, точка), (точка, прямая), (прямая, прямая), (точка, плоскость),
    - находить точку пересечения трёх попарно непараллельных плоскостей
    - находить объем тетраэдра.

ПРИЛОЖЕНИЕ Д

Министерство образования и науки РФ
Рубцовский индустриальный институт (филиал)
ФГБОУ ВПО «Алтайский государственный технический университет
им. И.И. Ползунова»

ПАМЯТКА
для студентов специальности ИВТ
по изучению дисциплины «Программирование» (Си)
(2 семестр)

Составила:_____
к.п.н., доцент кафедры ПМ
Н.А.Ларина
“____”_____ 2010г.

Утверждаю: _____
Зав. кафедрой ПМ, к.ф-м.н.,
доцент Е.А.Дудник
“____”_____2010г

1. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

Форма итоговой аттестации: **курсовая работа, экзамен.**

В **теоретическом курсе** будут рассмотрены следующие темы:

	Модуль 1
Лекция 1.	**Введение в алгоритмический язык СИ (2 ч).** История языка Си и его особенности. Структура программы Си. Структура программы Си. Константы.
Лекция 2.	**Простые типы данных (4 ч).** Целый, плавающий, символьный типы. Перечислимый тип (перечисления). **Операции языка СИ.**
Лекция 3.	**Операторы СИ (6ч).** Условные операторы. Операторы цикла. Вспомогательные управляющие операторы.
	Модуль 2
Лекция 4.	**Массивы и указатели. Сложные структуры данных (4ч).** Классы памяти. Массивы. Указатели. Структуры (записи). Объединения (смеси).
Лекция 5.	**Препроцессор и макросы (2ч)** Макроопределения и препроцессор. Условная компиляция. Прагмы.
	Модуль 3
Лекция 6.	**Организация ввода-вывода (6ч).** Ввод-вывод трех уровней. Ввод-вывод нижнего уровня. Функции ввода-вывода верхнего уровня (потока). Управление буферизацией. Двоичный и текстовый режимы. Ввод-вывод для консоли и порта. Форматизованный ввод-вывод. Функции обработки строк.
	Модуль 4
Лекция 7.	**Функции в СИ. Динамические объекты (4ч).** Функции. Передача параметров. Организация проекта. Указатели. Управление динамической памятью. Динамически связанные списки.
Лекция 8.	**Графика (4ч).** Инсталляция графического режима. Функции построения фигур. Перемещение фигур по экрану. Использование окон.

Лабораторные работы

	Модуль 1
Занятие 1	Лабораторная работа №1. Структура программы СИ. (2ч)
Занятие 2	Лабораторная работа №2. Основные операции (2ч)
Занятие 3	Лабораторная работа №3. Операторы СИ (2ч)

Занятие 4	Лабораторная работа №4. Операторы СИ (2ч)
	Модуль 2
Занятие 5	Лабораторная работа №5. Массивы (2ч)
Занятие 6	Лабораторная работа №6. Массивы (2ч)
Занятие 7	Лабораторная работа №7. Структуры (2ч)
Занятие 8	Лабораторная работа №8. Объединения (2ч)
Занятие 9	Лабораторная работа №9. Препроцессор и организация макросов (2ч)
	Модуль 3
Занятие 10	Лабораторная работа №10. Текстовые файлы (2ч)
Занятие 11	Лабораторная работа №11. Прямой доступ (2ч)
	Модуль 4
Занятие 12	Лабораторная работа №12. Функции (2ч)
Занятие 13	Лабораторная работа №13. Работа с динамическими структурами (2ч)
Занятие 14	Лабораторная работа №14. Организация проекта (2ч)
Занятие 15	Лабораторная работа №15. Организация проекта (2ч)
Занятие 16	Лабораторная работа №16. Графика (2ч)
Занятие 17	Лабораторная работа №17. Графика (2ч)

Практические занятия

	Модуль 1
Занятие 1-2	Практическое занятие №1. Структура программы СИ. (4ч)
Занятие 3-4	Практическое занятие №2. Операции языка СИ. (4ч)
Занятие 5-6	Практическое занятие №3. Основные операторы (4ч)
	Модуль 2
Занятие 7-8	Практическое занятие №4. Одномерные массивы (4ч)
Занятие 9-10	Практическое занятие №5. Строки. Матрицы (4ч)
Занятие 11-12	Практическое занятие №6. Структуры, объединения (4ч)
	Модуль 3
Занятие 13-14	Практическое занятие №7. Работа файлами (2ч)
	Модуль 4
Занятие 15-16	Практическое занятие №8. Функции, динамические структуры (4ч)
Занятие 17	Итоговое (2ч.)

2. УЧЕБНО-МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ

1. Архангельский А.Я. Язык C++ в C++Builder: Справ. и метод. пособие [текст] / А.Я. Архангельский.- М.: БИНОМ, 2008. -944 с.
2. Джехани Н. Программирование на языке СИ. М.: Мир, 1998.
3. Керниган Б., Ричи Д. Язык программирования СИ. – М.: Мир, 1985.
4. Самюэл П. Язык программирования С: [текст]: Пер. с англ/ П. Самюэл Л. Гай. -М.: Бином-Пресс, 2009. -528 с.
5. Сахань Ю.В. Задания для курсовых работ по алгоритмическим языкам: Методические указания и задания по выполнению курсовых работ для студентов специальности 073000 «Прикладная математика» Рубцовский индустриальный институт.- Рубцовск, 2006.– 34с.
6. Сахань Ю.В. Языки программирования. Часть 2 – СИ: Методические указания и задания к лабораторным работам для студентов специальности 073000 «Прикладная математика» Рубцовский индустриальный институт.- Рубцовск, 2006.– 41с.
7. Фигурнов В.Э. IBM PC для пользователя. – М.: Финансы и статистика, 1997.
8. Эпштейн М.С. Практикум по программированию по С: [текст]: Учеб. пособие для ссузов/ М.С. Эпштейн. -М.: Академия, 2007. -128 с.
9. http://www.helloworld.ru/texts/comp/lang/c/c5/index.htm
10. http://mark.zlatoust.ru/comp/lang/c_guide.html
11. http://lib.ru/MAN/DEMOS210/c.txt

12. *Керниган Б., Ритчи Д.* Язык программирования Си = The C programming language. - 2-е изд. - М.: Вильямс, 2007. -304с.
13. *Герберт Шилдт.* C: полное руководство, классическое издание = C: The Complete Reference, 4th Edition.— М.: Вильямс, 2010. -704с.
14. *Прата С.* Язык программирования C: Лекции и упражнения = C Primer Plus. - М.: Вильямс, 2006. -960с.
15. *Кочан С.* Программирование на языке Си = Programming in C. -3-е изд. - М.: Вильямс, 2006. -496.
16. *Гукин Д.* Язык программирования Си для «чайников» = C For Dummies. - М.: Диалектика, 2006. -352с.
17. *Axel-Tobias Schreiner.* Object oriented programming with ANSI-C. -Hanser, 2011. -223p.
18. ISO/IEC JTC1/SC22/WG14 official home (англ.). - Официальная страница международной рабочей группы по стандартизации языка программирования Си. Архивировано из первоисточника 22 августа 2011.
19. WG14 N1124 (англ.). *ISO/IEC 9899 — Programming languages — C — Approved standards*. ISO/IEC JTC1/SC22/WG14 (6 мая 2005).— Стандарт ISO/IEC 9899:1999 (C99) + ISO/IEC 9899:1999 Cor. 1:2001(E) (TC1 — Technical Corrigendum 1 от 2001 года) + ISO/IEC 9899:1999 Cor. 2:2004(E) (TC2 — Technical Corrigendum 2 от 2004 года).
20. C — The ISO Standard — Rationale, Revision 5.10 (англ.) (апрель 2004).— Обоснование и пояснения для стандарта C99.
21. Учебники по программированию, а также по C и C++ в библиотеке Максима Мошкова

3. ГРАФИК КОНТРОЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТА

Модуль	Контрольное испытание	Время проведения	Вес в итоговом рейтинге	Примечания
1	Посещение и активность на лекции		0,1	Оцениваются по 100 баллов
	Практическое занятие №1	по распис.занятий	0,03	
	Практическое занятие №2	по распис.занятий	0,03	
	Практическое занятие №3	по распис.занятий	0,03	
2	Практическое занятие №4	по распис.занятий	0,03	
	Практическое занятие №5	по распис.занятий	0,03	
	Практическое занятие №6	по распис.занятий	0,03	
3	Практическое занятие №7	по распис.занятий	0,03	
4	Практическое занятие №8	по распис.занятий	0,03	
1	Защита лаб.работы №1	по распис.занятий	0,015	Оцениваются по 100 баллов
	Защита лаб.работы №2	по распис.занятий	0,015	
	Защита лаб.работы №3	по распис.занятий	0,015	
	Защита лаб.работы	по	0,015	

2	№4	распис.занятий		
	Защита лаб.работы №5	по распис.занятий	0,015	
	Защита лаб.работы №6	по распис.занятий	0,015	
	Защита лаб.работы №7	по распис.занятий	0,015	
	Защита лаб.работы №8	по распис.занятий	0,015	
3	Защита лаб.работы №9	по распис.занятий	0,015	
	Защита лаб.работы №10	по распис.занятий	0,015	
	Защита лаб.работы №11	по распис.занятий	0,015	
	Защита лаб.работы №12	по распис.занятий	0,015	
	Защита лаб.работы №13	по распис.занятий	0,02	
4	Защита лаб.работы №14	по распис.занятий	0,015	
	Защита лаб.работы №15	по распис.занятий	0,015	
	Защита лаб.работы №16	по распис.занятий	0,015	
	Защита лаб.работы №17	по распис.занятий	0,015	
Курсовая работа		сессия	0,2	Оценивается по 100 баллов
Экзамен		сессия	0,2	2 вопроса: теорет. – 40 баллов, практич. – 60 баллов

Примечания:

1. Любая контрольная точка, выполненная после срока без уважительной причины, оценивается на 10% ниже. Максимальная оценка в этом случае 90 баллов.

2. Задолженности, ликвидированные во время сессии, оцениваются на 25% ниже. Максимальная оценка в этом случае 75 баллов.

3. За практическое занятие студент получает баллы в том случае, если он решал задачу у доски.

4. Курсовая работа, выполненная после сессии, оценивается на 30% ниже. Максимальная оценка в этом случае 70 баллов.

5. В конце семестра экзамен по дисциплине проводится в машинном зале. На подготовку отводится 40 минут. Студент получает индивидуальный билет, в котором содержится два вопроса – теоретический и практический.

4. ШКАЛА ОЦЕНОК И ПРАВИЛА ВЫЧИСЛЕНИЯ РЕЙТИНГА

В РИИ АлтГТУ принята **100-бальная** система оценок. Именно эти оценки учитываются при подсчете рейтингов, назначении стипендии и в других случаях. Традиционная шкала будет использоваться только в зачетных книжках. Соответствие оценок устанавливается следующим образом: 75 баллов и выше – «**отлично**», 50-74 балла – «**хорошо**», 25-49 баллов – «**удовлетворительно**», менее 25 баллов – «**неудовлетворительно**» .

Успеваемость студента оценивается с помощью текущего рейтинга (во время каждой аттестации) и итогового рейтинга (после сессии). Во всех случаях рейтинг вычисляется по формуле: $R_{итог} = \frac{\sum R_i p_i}{\sum p_i}$, где R_i – оценка за i-ю контрольную точку, p_i – вес этой контрольной точки. Суммирование проводится по всем контрольным точкам с начала семестра до момента вычисления рейтинга.

За посещение лекций и активное участие на них студент получает дополнительные баллы, которые определяются по следующей схеме:

Посещ ≤ 50% – Бп=0
50% ≤ Посещ ≤ 60% – Бп=2
60% ≤ Посещ ≤ 70% – Бп=4
70% ≤ Посещ ≤ 80% – Бп=6
80% ≤ Посещ ≤ 90% – Бп=8
90% ≤ Посещ ≤ 100% – Бп=10

Критерии оценивания практических заданий

0-24 баллов	Студент не решает задачу, не отвечает на вопросы.
25-49 баллов	В решении задачи допущено более двух логических ошибок (студент путает синтаксис основных операторов). Студент затрудняется отвечать на вопросы.
50-74 баллов	В решении задачи имеются недочеты, одна-две логические ошибки. Студент отвечает не на все вопросы.
75-100 баллов	Задача решена верно. В оформлении допускаются синтаксические ошибки. Студент отвечает на вопросы по программе.

Критерии оценивания лабораторных работ

0-24 баллов	Лабораторная работа выполнена частично. Студент не поясняет код написанной программы и не отвечает на вопросы.
25-49 баллов	Лабораторная работа выполнена верно, не в полном объеме. Студент путается при объяснении написанной программы, затрудняется отвечать на вопросы.
50-74 баллов	Лабораторная работа выполнена верно, в полном объеме. Студент поясняет не все операторы программы, отвечает не на все вопросы к лабораторной работе.
75-100 баллов	Лабораторная работа выполнена верно, в полном объеме. Студент поясняет код написанной программы, отвечает на все вопросы к лабораторной работе.

Критерии оценивания курсовой работы

0-24 баллов	Курсовая работа выполнена не в полном объеме. Документация не оформлена.
25-49 баллов	Курсовая работа выполнена не в полном объеме, сдана в установленные сроки. Имеются расхождения между техническим заданием и готовым программным продуктом. Документация оформлена с нарушениями требований ГОСТа.
50-74 баллов	Курсовая работа выполнена не в полном объеме, сдана в установленные сроки. Имеются небольшие расхождения между техническим заданием и готовым программным продуктом. Документация оформлена согласно ГОСТам.
75-100 баллов	Курсовая работа выполнена в полном объеме, сдана в установленные сроки. Программа написана в соответствии с техническим заданием. Документация оформлена согласно ГОСТам.

Пример расчета рейтинга:

Пусть студент получил следующие оценки:

Модуль 1: пр.зан.№1 – 26 б, пр.зан.№2 – 30 б, пр.зан.№3 – 48 б

Модуль 2: пр.зан.№4 – 60 б, пр.зан.№5 – 41 б, пр.зан.№6 – 30 б
Модуль 3: пр.зан.№7 – 30 б
Модуль 4: пр.зан.№8 – 30 б
Модуль 1: лаб.раб.№1 – 53 б, лаб.раб.№2 – 61 б, лаб.раб.№3 – 58 б лаб.раб.№4 – 58 б.
Модуль 2: лаб.раб.№5 – 72 б, лаб.раб.№6 – 40 б, лаб.раб.№7 – 80 б, лаб.раб.№8 – 58 б. лаб.раб.№9 – 58 б.
Модуль 3: лаб.раб.№10 – 63 б, лаб.раб.№11 – 72 б.
Модуль 4: лаб.раб.№12 – 63 б, лаб.раб.№13 – 72 б. лаб.раб.№14 – 58 б. лаб.раб.№15 – 58 б. лаб.раб.№16 – 63 б. лаб.раб.№17 – 72 б.
Посетил 85% лекций
Курсовая работа: 56 баллов.
Экзамен: за ответ на экзамене студент получил 54 балла.
На аттестации(10 неделя) его текущий рейтинг равен:

$$R_{атт} = \frac{0{,}03 \cdot (26+30+48+60+41) + 0{,}015 \cdot (53+61+58+58+72+40+80+58+58+63)}{0{,}03 \cdot 5 + 0{,}015 \cdot 10} = 51$$

Перед началом сессии вычисляется семестровый рейтинг:

$$R_{атт} = \frac{0{,}03 \cdot (26+30+48+60+41+30+30+30) + 0{,}015 \cdot (53+61+58+58+72+40+80+58+58+63+72+63+72+58+58+632) + 72 \cdot 0{,}02 + 0{,}2 \cdot 56}{0{,}03 \cdot 8 + 0{,}015 \cdot 16 + 0{,}02 + 0{,}2} = 52$$

Итоговый рейтинг вычисляется после экзамена по формуле:

$$R_{итог} = R_{сем} \cdot 0{,}7 + R_{экз} \cdot 0{,}2 + Б_{П} = 52 \cdot 0{,}7 + 54 \cdot 0{,}2 + 8 = 55$$

В экзаменационную ведомость и зачетку выставляется оценка «**хорошо**» и **55** баллов. За курсовую работу выставляется оценка «**хорошо**» и **56** баллов.

ПРАВИЛА ДОПУСКА К ЗАЧЕТУ И ВЫСТАВЛЕНИЕ «АВТОМАТОВ»

- К экзамену допускаются студенты, защитившие **курсовую работу** и имеющие $R_{сем} \geq$ **25** баллов.
- Студент получает экзамен, если его ответ на экзамене **≥50** баллов и $R_{сем} \geq$ **30.**
- Студент получает экзамен «**автоматом**», если все самостоятельные работы сданы в установленные сроки, есть не менее 2 ответов на практических занятиях и семестровый рейтинг **≥70**.
- Оценка за курсовую работу выставляется согласно установленным критериям.

ПРИЛОЖЕНИЕ Е

Задания для лабораторных занятий

Лабораторная работа 1.

Изучение системы Borland – C

1. Войти в систему, изучить все режимы главного меню.
2. Набрать в окне редактора программу, которая в виде таблицы распечатает размеры памяти для переменных различных типов и их модификаций (не менее 10), используя функцию SIZEOF().
3. Выполнить компиляцию программы, исправить ошибки и запустить на выполнение.
4. Прокомментировать строки программы и уметь объяснить каждую.
5. Подготовить ответы на темы:
   a. Структура Си программы.
   b. Лексемы.
   c. Константы.

Лабораторная работа 2.

Основные операторы Си

1. Составьте программу:

В-1. а) Последовательность N целых чисел вводится с терминала. Вычислить отдельно среднее арифметическое ее нечетных элементов и среднее арифметическое ее четных элементов.

б) Вычислите среднее арифметическое наперед незаданного количества чисел.

В-2. а) Найдите все делители некоторого натурального числа.

б) Разложите целое число на простые множители, которые выведите на экран в порядке возрастания.

В-3. а) Написать программу, которая вычисляет сумму цифр данного целого числа.

б) Дано несколько целых чисел. Определить, сколько среди них четных чисел.

В-4. а) Вычислить (n)!!, где (2n)!!=2*4*…*(2n)
(2n+1)!!=1*3*…*(2n+1).

б) Определить первую и последнюю цифру данного целого числа.

В-5. а) Составить программу, проверяющую, будет ли простым данное натуральное число.

б) Вычислить (1+sin 0.1)(1+ sin 0.2)…(1+ sin 10).

В-6. а) Даны натуральное n, действительное X. Вычислить: sin X + sin sin X+ sin sin…sin X.

б) Дано целое n>2. Напечатать все простые числа из диапазона [2,n].

В-7. а) Билет считается «счастливым», если сумма трех его старших цифр равна сумме трех младших. Определить, будет ли билет с данным номером «счастливым».

б) Составить таблицу значений функции $y = \frac{e^x}{\sqrt{1+ax^2}}$ на отрезке [-1;3] с шагом изменения аргумента 0,25, где a=3,2.

В-8. а) Последовательность N целых чисел вводится с терминала. Определить минимальный элемент из данных.

б) Дано целое n>2. Напечатать все простые числа из диапазона [2,n].

В-9. а) Найдите все делители некоторого натурального числа.

б) Определить первую и последнюю цифру данного целого числа.

В-10. а) Последовательность N целых чисел вводится с терминала. Вычислить отдельно среднее арифметическое ее нечетных элементов и среднее арифметическое ее четных элементов.

б) Написать программу, которая вычисляет сумму цифр данного целого числа.

В-11. а) Даны натуральное n, действительное X. Вычислить: sin X + sin sin X+ sin sin…sin X.

б) Дано несколько целых чисел. Определить, сколько среди них четных чисел.

В-12. а) Билет считается «счастливым», если сумма трех его старших цифр равна сумме трех младших. Определить, будет ли билет с данным номером «счастливым».

б) Разложите целое число на простые множители, которые выведите на экран в порядке возрастания.

2. Войдите в систему и наберите текст программы.
3. Откомпилируйте программу и запустите ее на выполнение.
4. Объясните каждую строчку программы.
5. Подготовьте ответ по следующим темам:
   - Типы данных.
   - Операции.

Лабораторная работа №3
Основные операторы С

1. Составьте программу приближенного вычисления функции с заданной точностью eps с помощью ряда. На экран выводить промежуточные значения (значение каждого слагаемого) и конечный результат.

В-1. $y = \sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$

В-2. $y = \cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$

В-3. $y = \exp(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$

В-4. $y = sh(x) = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \ldots$

В-5. $y = ch(x) = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \ldots$

В-6. $y = \ln(x+1) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots$ в интервале (-1,1)

В-7. $y = (x+1)^n = 1 + nx + \left(\frac{n(n-1)}{2!}\right)x^2 + \left(\frac{n(n-1)(n-2)}{3!}\right)x^3 \ldots$ в интервале (-1,1)

В-8. $y = arctg(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$

В-9. $y = \exp(-x) = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \ldots$

В-10. $y = (x+1)^n = 1 + nx + \left(\frac{n(n-1)}{2!}\right)x^2 + \left(\frac{n(n-1)(n-2)}{3!}\right)x^3 \ldots$ в интервале (-1,1)

В-11. $y = \sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$

В-12. $y = \cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$

2. Войдите в систему и наберите текст программы.
3. Откомпилируйте программу и запустите ее на выполнение.
4. Объясните каждую строчку программы.
5. Подготовьте ответ по следующим темам:
   - Условные операторы.
   - Операторы цикла.
   - Вспомогательные управляющие операторы.

Лабораторная работа №4
Работа с массивами

1. Составьте программу:

В-1. а) Дана последовательность N элементов в виде массива. Посчитать сумму его положительных элементов с нечетными номерами.

б) Даны две матрицы A и B размерности m*n. Найти произведение A*B.

В-2. а) Дана последовательность N элементов в виде массива. Вычислить среднее арифметическое положительных элементов с четными номерами.

б) Дана матрица A размерности m*n и вектор X размерности n. Найти произведение A*X.

В-3. а) Дана последовательность N элементов в виде массива. Найти количество элементов с нечетными номерами, которые больше последнего.

б) Дана матрица B размерности m*n. Найти столбец с минимальной суммой элементов и выдать этот элемент на терминал.

В-4. а) Дана последовательность N элементов в виде массива. Найти минимальный элемент массива, поставить его на первое место, остальные элементы сдвигаются.

б) Дана матрица A размерности m*n и вектор X размерности m. Найти произведение X*A.

В-5. а) Дана последовательность N элементов в виде массива. Найти все элементы, которые удовлетворяют условию A[i]=2*i, и их количество.

б) Дана матрица A размерности m*n. Найти сумму наименьших значений ее столбцов.

В-6. а) Дана последовательность N элементов в виде массива. Найти сумму тех элементов, которые удовлетворяют условию |A[i]|=i*3.

б) Дана матрица A размерности m*n. Найти строку с максимальной сумой элементов и выдать ее на терминал.

В-7. а) Дана последовательность N элементов в виде массива. Написать программу для уменьшения всех его положительных компонент в 2 раза.

б) Дана матрица A размерности n*n. Проверить, является ли матрица магическим квадартом.

В-8. а) Дана последовательность N элементов в виде массива. Выписать все элементы, которые больше первого и меньше последнего, если они есть, иначе выдать сообщение об их отсутствии.

б) Дана матрица A размерности n*n. Найти в каждой строке минимальный элемент и поставить его на место диагонального этой строки.

В-9. а) Дана последовательность N элементов в виде массива. Найти сумму и количество тех его элементов, которые делятся на 5 и не делятся на 3.

б) Дана матрица A размерности m*n. Найти сумму наибольших значений элементов ее строк.

В-10. а) Дана последовательность N элементов в виде массива. Найти количество элементов с нечетными номерами, которые больше последнего.

б) Дана матрица A размерности m*n. Найти строку с максимальной сумой элементов и выдать ее на терминал.

2. Войдите в систему и наберите текст программы.
3. Откомпилируйте программу и запустите ее на выполнение.
4. Объясните каждую строчку программы.
5. Подготовьте ответ по следующим темам:
   - Классы памяти.
   - Массивы.
   - Указатели.

Лабораторная работа №5
Работа со структурами в С

1. Составьте программу, организации справочника жителей города, используя структуру:

```
struct {
```

```
        char fam[16];     /*фамилия*/
        char im[10];      /*имя*/
        char ot[12];      /*отчество*/
        char ul[16];      /*улица*/
        unsigned nom;         /*дом*/
        unsigned kw;          /*квартира*/
    }
```

2. Войдите в систему и наберите текст программы.
3. Откомпилируйте программу и запустите ее на выполнение.
4. Объясните каждую строчку программы.
5. Подготовьте ответ по следующим темам:
   – Структуры (записи).
   – Объединения (смеси).

Лабораторная работа №6
Работа с объединением в С

1. Составьте программу распечатки размеров структуры, используя объединение:

```
struct {
        char fam[16];
        double x, y;
        int i, j;
        union {
                char *c;
                int *k;
                int *kw;}a;
        struct {
                unsigned f1:2;
                unsigned f2:4;
                unsigned :0;
                unsigned f3:8;} b;
} c;
```

   с помощью функции sizeof();
2. Войдите в систему и наберите текст программы.
3. Откомпилируйте программу и запустите ее на выполнение.
4. Объясните каждую строчку программы.

Лабораторная работа №7
Ввод-вывод

1. Написать программу создания файла в двоичном виде, содержащего натуральные числа. Использовать верхний уровень.

2. Написать программу, в которой находится среднее чисел файла, созданного в предыдущем пункте.
3. Войдите в систему и наберите тексты программ.
4. Откомпилируйте программу и запустите ее на выполнение.
5. Объясните каждую строчку программы.
6. Подготовьте ответ по следующим темам:
   – Ввод-вывод трех уровней.
   – Ввод-вывод нижнего уровня.
   – Ввод-вывод потока.

Лабораторная работа №8
Прямой доступ

1. Написать программу создания файла прямого доступа, содержащего сведения о числе деталей с заданным кодом. Структура записи:

```
typedef struct {
        int nom;          /*номер*/
        char name[10];    /*наименование*/
        unsigned kol;     /*количество*/
        } spis;
```

   В программе создается файл прямого доступа и организуется просмотр на терминале записи по ее номеру.
2. Войдите в систему и наберите текст программы.
3. Откомпилируйте программу и запустите ее на выполнение.
4. Объясните каждую строчку программы.
5. Подготовьте ответ по следующим темам:
   – Управление буферизацией.
   – Двоичный и текстовый режимы.
   – Ввод-вывод для консоли и порта.
   – Форматизованный ввод-вывод.
   – Функции обработки строк.

Лабораторная работа №9
Препроцессор и организация макросов в С

1. Составьте программу, в которой задана MACRO для вычисления степени, головная программа вводит число, возводимое в степень, и показатели степени:

   a) с использованием стандартной функции возведения в степень;
   b) без использования стандартной функции.
2. Войдите в систему и наберите текст программы.
3. Откомпилируйте программу и запустите ее на выполнение.
4. Объясните каждую строчку программы.

5. Подготовьте ответ по следующим темам:
   – Макроопределения и препроцессор.
   – Условная компиляция.
   – Прагмы.

Лабораторная работа №10
Функции в С.
Сортировка массива методом «Пузырька»

1. Написать программу сортировки массива методом «Пузырька». Сортировку оформить функцией, а в головной программе выполнить ввод массива.
2. Войдите в систему и наберите текст программы.
3. Откомпилируйте программу и запустите ее на выполнение.
4. Объясните каждую строчку программы.
5. Подготовьте ответ по следующим темам:
   – Понятие функции.
   – Определение, объявление и вызов функции.
   – Передача параметров.

Лабораторная работа №11
Организация проекта в С.
Пакет подпрограмм

1. Составьте пакет работы с комплексными числами. Наберите файл исполняющих процедур, которые должны выполнять действия над комплексными числами a+jb:

   Сложение add – a1+a2 действ. часть
   b1+b2 мним. часть
   Вычитание sub – a1–a2 действ. часть
   b1–b2 мним. часть
   Умножение mult – a1*a2–b1*b2 действ. часть
   b1*a2+a1*b2 мним. часть
   Деление div – (a1*a2+b1*b2)/(a2*a2+b2*b2) действ. часть
   (a1*b1+a1*b2)/(a2*a2+b2*b2) мним. часть

   Предусмотреть следующие операции:
   a) Присвоение действительной части комплексному числу;
   b) Присвоение мнимой части комплексному числу;
   c) Распечатка на терминал комплексного числа.
2. Войдите в систему и наберите текст программы.
3. Откомпилируйте программу и запустите ее на выполнение.
4. Объясните каждую строчку программы.

Лабораторная работа №12
Организация стека с помощью списковой структуры

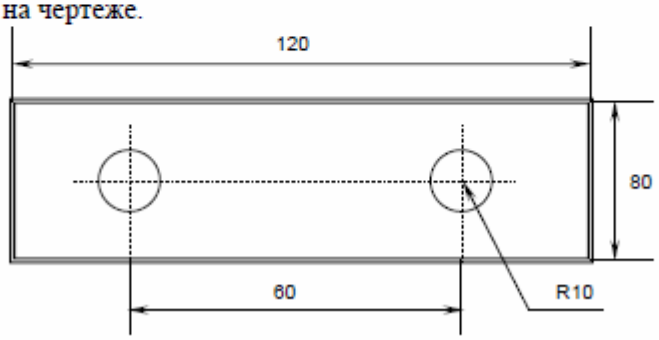
1. Написать программу организации калькулятора, работающего со стековой структурой по принципу обратной польской последовательности. Стек организован в виде списковой структуры, загрузку в стек и выгрузку элементов оформить процедурами. Элементы-числа в контрольном примере разделены пробелами, элементы-знаки – знаками табуляции. Ввод данных из файла на диске, вывод результата на терминал.
2. Войдите в систему и наберите текст программы.
3. Откомпилируйте программу и запустите ее на выполнение.
4. Объясните каждую строчку программы.
5. Подготовьте ответ по следующим темам:
   – Указатели.
   – Управление динамической памятью.
   – Функции управления памятью.

Лабораторная работа №13
Графический режим

1. Установите графический режим.
2. Создайте окружность, перемещающуюся по экрану. При движении в разные стороны изменять цвет фона, внутреннюю закраску и шаблон заполнения.
3. Войдите в систему и наберите текст программы.
4. Откомпилируйте программу и запустите ее на выполнение.
5. Объясните каждую строчку программы.
6. Подготовьте ответ по следующим темам:
   – Инсталляция графического режима.
   – Установка курсора, цвета, стиля, палитры.
   – Функции вычерчивания линий и замкнутых фигур.

Лабораторная работа №14
Графический режим

1. Составить программу построения чертежа детали «ПЛАНКА». Цвет закраски должен отличаться от цвета фона, закраска сплошная. Проставить размеры на чертеже.



2 Войти в систему и набрать текст программы.

3 Компилируйте программу и запустите её на выполнение.

4 Комментируйте программу и умейте объяснить каждую её строку.

5 Подготовить ответы на темы:

- Способы перемещения изображения по экрану.
- Использование окон.

ПРИЛОЖЕНИЕ Ж

Задания для курсовых работ

Оформление и состав курсовой работы должны соответствовать курсовой работе в первом семестре по предмету «Программирование» часть 1 (Паскаль).

Титульный лист и задание в отчёте оформляется согласно ГОСТу по оформлению управленческой документации смотрите в методических указаниях к предмету «Деловая корреспонденция».

Ниже перечислена литература из общего списка (с.), которая упомянута в заданиях для курсовых работ:

1. Баглаев Ю.П. Вычислительная математика и программирование. –М.: Высш.шк., 1990. -544с.
2. Воробьёв Г.Н., Данилова А.Н. Практикум по вычислительной математике. –М.: Высш.шк., 1990. -208с.
3. Дьяконов В.П. Справочник по алгоритмам и программам на языке Бейсик для персональных ЭВМ. –М.: Наука, 1987. -240с.
4. Шуп Т.Е. Прикладные численные методы в физике и технике. –М.: Высш.шк., 1990. -400с.

Варианты заданий:

3.1 Решение системы линейных уравнений методом Жордана-Гаусса с выбором ведущего элемента

Метод основан на приведении системы линейных уравнений

$$a_{11}x_{1+}a_{12}x_{2+}\dots+\ a_{1n}x_n=c_1$$

$$a_{21}x_{2+}a_{22}x_{2+}\dots+\ a_{2n}x_n=c_2$$

$$\dots\dots\dots\dots\dots$$

$$a_{n1}x_{2+}a_{n2}x_{2+}\dots+\ a_{nn}x_n=c_n$$

к треугольному виду.

При применении метода Гаусса первое главное уравнение делят на a_{11}, затем первое уравнение последовательно умножают на коэффициент a_{11} других уравнений и вычитают из остальных уравнений. В результате первая переменная будет исключена из всех уравнений, кроме первого. Далее этот процесс применяют к остальным $n-1$ уравнениям для исключения второй неизвестной. Эта процедура применяется, пока система не будет приведена к треугольному виду.

На k-м шаге коэффициенты k -го уравнения имеют вид:

$$b_{kj} = a_{kj} / a_{kk},$$

при этом коэффициенты других уравнений:

$$b_{kj} = a_{ij} - a_{ik*} a_{kj} / a_{kk}.$$

Элемент a_{kk} называется ведущим (разрешающим). Важен выбор ведущего элемента, чтобы он был отличен от нуля. Можно показать, что наибольшая точность достигается, когда ведущий элемент имеет наибольшее значение. Алгоритм этого метода можно найти в литературе и Интернете.

Таблица 1 – Варианты заданий

Номер задания	a_{i1}	a_{i2}	a_{i3}	a_{i4}	a_{i5}	Входной файл	Выходной файл
3.1.1	-2 3,2 3,4 2,6	1,1 2,1 2,3 1,1	-2,0 3,2 4,1 -3,2	-1,8 2,2 3,2 2,4	1,0 1,0 6,0 -7,0	Задается редактором на диске	Вывод на терминал
3.1.2	1 2 -1	-2 3 -1	3 1 1	0 0 0	1 2 3	Вывод с терминала	Вывод на печать
3.1.3	4 0,09 0,04	0,24 3 -0,08	-1,1 -0,15 4	0 0 0	8 9 20	Файл создается программно на диске	Вывод на терминал
3.1.4	3 1 1	2 1 -2	1 -1 1	0 0 0	4 1 3	Файл двоичный создается программно	Вывод на печать
3.1.5	1,1 1 2,1 3	1 -1 1 1	1 -1 -1 2	-1 1 2 1	2 0 9 7	Создается редактором на диске	Вывод на печать

Составить программу решения системы уравнений, результаты вывести в виде таблицы.

РЕШЕНИЕ СИСТЕМЫ УРАВНЕНИЙ МЕТОДОМ _____				
x_1	x_2	x_3	x_4	x_5

3.2 Определение корней полинома

Линейное алгебраическое уравнение имеет вид

$$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 = 0 . \qquad (1)$$

При отыскании корней следует иметь в виду свойства:

1. Алгебраическое уравнение порядка n имеет n корней, которые могут быть действительными и комплексными.
2. Если коэффициент a_i действительное, то комплексные корни образуют комплексно-сопряженные пары.
3. Число положительных действительных корней равно или меньше числа перемен знаков в коэффициентах a_i.

Большинство методов решения связаны с выделением квадратичного множителя $x^2 + px + q$.

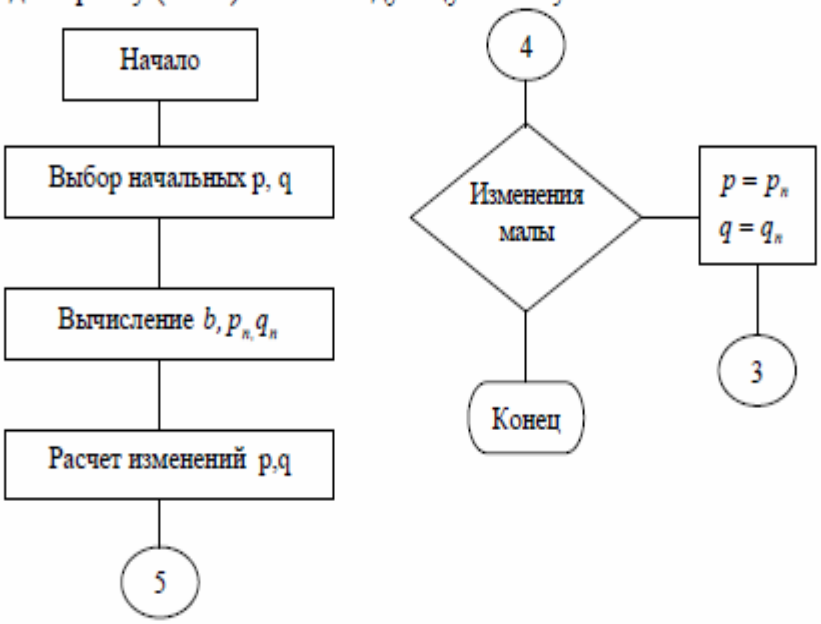
Метод Берстоу подобного типа рассмотрен в [5], метод Хичкока - в [4]. Алгебраическое уравнение (1) представляют в виде:

$$(x^2 + px + q)(x^{n-2} + b_{n-1}x^{n-3} + \ldots + b_3 x + b_2) + b_1 x + b_0 = 0. \quad (2)$$

Здесь $b_1 x\ b_0$ - линейный остаточный член, который стремится к нулю, коэффициенты p,q вычисляются приближенным методом. Если $b_1 и b_0$ близки к нулю, то, раскрыв в (2) скобки и сравнивая коэффициенты уравнений (1) и (2), получим систему:

$$\begin{aligned} B_{n-1} &= A_{n-1} - P \\ B_{n-2} &= A_{n-2} - PB_{n-1} - q \\ &\ldots\ldots\ldots\ldots \\ B_{n-j} &= A_{n-j} - QB_{n+1-j} - QB_{n-2-j} \\ &\ldots\ldots\ldots\ldots \\ P &= (A_1 - QB_3)/B_2 \\ Q &= A_0 / B_2 \end{aligned} \quad (3)$$

Алгоритм итерационной процедуры для выделения квадратного трехчлена в методе Берстоу (Лина) имеет следующую схему:



Далее процесс можно организовать следующим образом:

1. Вычисляются по обычным формулам решения квадратного уравнения корни трехчлена

$$x^2 + px + q = 0. \quad (4)$$

2. Вновь выделяется квадратный трехчлен итерационным методом по указанному выше алгоритму - и так до вычисления всех корней.

В файле содержится набор коэффициентов уравнения:

Таблица 2 - Варианты заданий

Номер задания	Уравнение	Точность	Входной файл	Выходной файл
3.2.1	$x^4 + 5x^3 + 11x^2 - 2x - 28 = 0$	0,001	Создан редактором на диске	Вывод на терминал
3.2.2	$x^4 + 3x^3 + 3x^2 = 0$	0,0001	Создан редактором на диске	Вывод на терминал
3.2.3	$x^4 + 4x^3 + 4x^2 + 4x - 1 = 0$	0,001	Создать программно на диске в виде двоичного файла	Вывод на терминал
3.2.4	$x^4 - 6x^2 - 12x - 8 = 0$	0,001	Ввод с терминала	Вывод на печать

Результат оформить таблицей:

ОПРЕДЕЛЕНИЕ КОРНЕЙ ПОЛИНОМА МЕТОДОМ_____				
x_1	x_2	x_3	x_4	Точность *eps*

3.3 Определение обратной матрицы

Обратной матрицей называется матрица такая, что произведение на заданную матрицу A дает единичную матрицу.

$$A \cdot A^{-1} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}. \quad (1)$$

Из этой системы уравнений можно найти элементы обратной матрицы A^{-1}. Решение системы линейных уравнений возможно прямым методом (например, методом Жордана-Гаусса, см. п. 3.1) или методами итераций, при этом систему приводят к виду:

$$\begin{aligned} x_1 &= b_{1n}x_n + b_{1n-1}x_{n-1} + \dots + b_{11}x_1 + b_{10} \\ x_2 &= b_{2n}x_n + b_{2n-1}x_{n-1} + \dots + b_{21}x_1 + b_{20} \\ &\dots\dots\dots\dots \\ x_n &= b_{nn}x_n + b_{nn-1}x_{n-1} + \dots + b_{n1}x_1 + b_{n0} \end{aligned} \quad (2)$$

В правой части выбирается вектор $(X_1, X_2, ... X_n)$ начального приближения и вычисляются новые значения неизвестных. В методе Гаусса-Зейделя уточненное значение X_1 сразу применяют при вычислении X_2, новые значения X_1, X_2 для вычисления X_3 и т.д. Алгоритм этого метода рассмотрен в [5].

В файле содержится набор элементов исходной матрицы A.

Таблица 3 – Варианты заданий

Номер задания	A_{i1}	A_{i2}	A_{i3}	A_{i4}	Входной файл	Выходной файл
3.3.1	1 6 2 7	2,1 7 3 8	3 8 4 9	4 9,2 5 1	Задается редактором	Вывод на терминал
3.3.2	1,1 6,3 2 7	2,5 7,2 3,1 8	3 4 8 9	4 8 1,1 1	Задается с терминала	Вывод на печать
3.3.3	2,1 6 2 7	1,1 7 3 8	3 3 4,2 9,1	4 10,1 16,3 1,1	Создается на диске программно в двоичном виде	Вывод на терминал
3.3.4	1,2 5 2 7	2,3 7 3 8	9 8 4 9	4 9,1 5,2 1,1	Создается реактором на диске	Вывод на печать

Составить программу нахождения обратной матрицы с использованием метода Гаусса-Зейделя. Результаты вывести в виде таблицы:

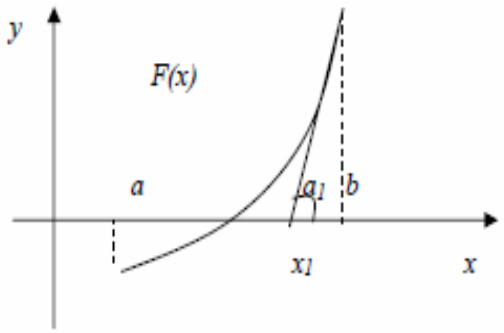
ОПРЕДЕЛЕНИЕ ОБРАТНОЙ МАТРИЦЫ МЕТОДОМ _____-го порядка			
x_{i1}	x_{i2}	x_{i3}	x_{i4}

3.4 Метод Ньютона для решения нелинейных уравнений

Решение нелинейных уравнений методом Ньютона для уравнения вида

$$F(x) = 0 \qquad (1)$$

на отрезке $[a,b]$ можно представить графически.



Из точки функции $F(x_0)$, где $x_0 = b$, проводят касательную. В результате получаем $x_1 = x_0 + dx$, где dx - поправка к начальному приближению x_0.

$$dx = F(x_0)/tg(a_1) = -F(x_0)/F'(x_0). \quad (2)$$

Общий вид итерационной формулы:

$$x_{n+1} = x_n - F(x_0)/F'(x_0). \quad (3)$$

Производная $F'(x)$ может быть вычислена приближенно:

$$F'(x) = (F(x_n + dx) - F(x_n)) \Big/ ((x_n + dx) - x_n). \quad (4)$$

В этом случае метод носит название модифицированного метода Ньютона. Если задана точность вычисления eps, то при $F(x_n) < eps$ вычисления прекращаются. Описание метода содержатся в [1], [4], [5], имеются готовые программы на Бейсик.

Таблица 4 – Варианты заданий

Номер задания	Уравнение	x_0	a	b	eps	Входной файл	Выходной файл
3.4.1	$y = x - \sin x - 0.25$		1,1	1,2	0,00001	Создается редактором на диске	Вывод на терминал
3.4.2	$y = \sin(x+1) - x - 1$				0,0001	Создается редактором на диске	Вывод на терминал
3.4.3	$y = \cos(x-1) + x - 0.8$				0,00001	Задается с терминала	Вывод на печать
3.4.4	$y = \cos(x+1) - x - 1.5$				0,0001	Создается программой в двоичном виде	Вывод на терминал

Результаты оформить в виде таблицы:

РЕШЕНИЕ НЕЛИНЕЙНЫХ УРАВНЕНИЙ МЕТОДОМ НЬЮТОНА		
Значения x	Точность	Количество шагов

3.5 Метод Ньютона для решения систем нелинейных уравнений

Метод основан на построении касательной, что связано с разложением функции в ряд Тейлора:

$$F(x_n + h) = F(x_n) + hF'(x_n) + (h^2/2)F''(x_n) + \ldots. \quad (1)$$

Члены, содержащие h во второй и выше степени, отбрасывают, предполагают, что переход от x_n к x_{n+1} приближают функцию к нулю, тогда при $x_{n+1} = x_{n+h}$

$$F(x_{n+h}) = 0. \quad (2)$$

Откуда

$$x_{n+1} = x_n - \mathrm{F}(\mathrm{x}_n)/F'(x_n). \quad (3)$$

Для определения производной применяют приближенный метод:

$$F'(x_n) = (F(x_n) - F(x_n - 1))/(x_n - x_{n-1}). \quad (4)$$

и метод в этом случае носит название модифицированного метода Ньютона. Описание метода для систем уравнений содержится в [1], [5], необходимо также изучить содержимое п. 3.4 настоящей работы. Составить программу решения системы уравнений модифицированным методом.

Таблица 5 – Варианты заданий

Номер задания	Уравнение	x_{10}	x_{20}	Точ-ность	Входной файл	Выходной файл
3.5.1	$x_1 + 3x_1x_2 - x_2 = 0$ $2x_1 - x_1x_2 - 5x_2 + 1 = 0$	3,4	2,2	10^{-4}	Создается редактором на диске	Вывод на печать
3.5.2	$x_1 = 0.25\sin(0.3x_2) + 2x_2 = 0.5\cos(0.5x_1)$	2	-3	10^{-3}	Ввод с терминала	Вывод на печать
3.5.3	$x_1 = 0.5\sin(0.3x_2) + 2x_2 = 0.5\cos(0.3x_1)$	0	0	10^{-4}	Создается редактором на диске	Вывод на терминал
3.5.4	$x_1 = 0.5\cos(0.2x_2) + 6x_2 = 0.5\cos(0.5x_1)$	2	-3,1	10^{-5}	Програм-мно на диске в виде двоичного файла	Вывод на печать

x_{10}, x_{20} – начальные приближения, задано также максимальное число итераций M=10 для 6, если решение расходится.

Отчет оформить в виде таблицы:

РЕШЕНИЕ СИСТЕМ НЕЛИНЕЙНЫХ УРАВНЕНИЙ МЕТОДОМ НЬЮТОНА			
x_1	x_2	*eps*	Число шагов

3.6 Решение систем линейных уравнений методом Холесского

Матрица коэффициентов системы уравнений:

$$A=\begin{vmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{21} & \dots & a_{2m} \\ \vdots & & & \vdots \\ & & & . \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{vmatrix} \quad (1)$$

преобразуется к эквивалентной верхней треугольной матрице.

Здесь в А последний столбец – правая часть системы уравнений. Преобразующая матрица нижняя треугольная, т.е. верно матричное равенство:

$$A=\begin{vmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{vmatrix} * \begin{vmatrix} 1 & u_{12} & u_{13} & u_{14} & u_{15} \\ 0 & 1 & u_{23} & u_{24} & u_{25} \\ 0 & 0 & 1 & u_{34} & u_{35} \\ 0 & 0 & 0 & 1 & u_{45} \end{vmatrix} = \begin{vmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \end{vmatrix} \quad (2)$$

После перемножения матриц найдем систему для определения элементов матриц. Первый столбец L:

$$l_{i1} = a_{i1}; \qquad i=1,2,3\ldots n. \quad (3)$$

Зная его, первая строка u найдется

$$u_{1j} = a_{1j}/l_{11}; \quad j=1,2,3,\ldots n+1. \quad (4)$$

Далее

$$l_{i2} = a_{i2} - l_{21} * u_{22}; \qquad i=1,2,3,\ldots n \quad (5)$$

$$u_{2j} = (a_{2j} - l_{21} * u_{1j})/l_{22}; \qquad j=1,2,3,\ldots n+1. \quad (6)$$

Для экономии оперативной памяти можно наложить матрицы U и L на А, тогда

$$A_{1j} = A_{1j} / A_{11}; \quad j=2,3,\ldots,n+1 \quad \text{для первой строки} \quad (7)$$

$$A_{im} = A_{im} - \sum_{k=1}^{m-1} A_{ik} * A_{im}; \; i=m,\ldots,n \text{ для очередного } L_{im} \text{ столбца}. \quad (8)$$

Для очередной U_{mj} строки

$$A_{mj} = \frac{A_{mj} - \sum_{k=1}^{m-1} A_{mk} * A_{kj}}{A_{mm}}; \quad j=m+1,\ldots,n+1 \quad . \quad (9)$$

После того, как вычислена матрица U, обратной подстановкой вычисляются x

$$x_n = A_{n,n+1}$$

$$x_i = A_{i,n+1} - \sum_{k=i+1}^{n} A_{ik} * x_k \; ; \qquad i=n-1,\ldots,1.$$

Ведущий элемент A_{mm} выбирается как наибольший.

Составить программу для решения систем методом Холесского [5].

Таблица 6 – Варианты заданий

Номер задания	a_{i1}	a_{i2}	a_{i3}	a_{i4}	a_{i5}	Входной файл	Выходной файл
3.6.1.	-2 3,2 3,4 2,6	1,1 2,1 2,3 1,1	-2,6 3,2 4,1 -3,1	-1,8 2,2 3,2 2,4	1 1 6 -7	Создается редактором на диске	Вывод на терминал
3.6.2.	0,63 1,17 3,58 2,71	1,0 0,18 0,21 0,75	0,71 -0,65 -3,45 1,17	0,34 0,71 -1,18 -2,35	2,08 0,17 0,05 1,28	Создается редактором на диске	Вывод на терминал
3.6.3.	1,0 0,33	-2,1 -0,77	2,04 0,44	0,17 -0,51	0,18 0,19	Ввод с терминала	Вывод на печать
3.6.4.	0,63 0,54 0,24 0,43	-0,7 0,88 -0,44 -1,2	1,34 -0,74 0,35 2,32	0,37 -1,27 0,55 -1,4	1,21 0,89 0,25 1,56	Программой создается двоичный файл на диске	Вывод на терминал

Решение оформить с помощью таблицы

РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ МЕТОДОМ ХОЛЕССКОГО				
x_1	x_2	x_3	x_4	Количество шагов

3.7 Нахождение наибольшего собственного значения матрицы

Общая формулировка задачи на собственное значение имеет вид

$$Ax = yx, \qquad (1)$$

где A – n*n матрица.

Ищется собственный вектор X и n скалярных величин y, удовлетворяющих этому уравнению. Если матрица A вещественна и симметрична, то все собственные числа y вещественные, что и бывает часто в инженерных задачах расчета на прочность.

Процедуру поиска наибольшего по модулю собственного значения начинают с выбора пробного нормированного вектора X(0). Этот вектор умножают на A, полученный вектор X представляют в виде произведения константы (собственного значения) и нормированного вектора X(1). Если X(0) и X(1) совпадают, то процедура окончена, в противном случае все повторяется снова. Нормирование вектора может быть проведено поиском наибольшего элемента и делением на него элементов вектора.

Сведения о методе содержатся в [3], [5]. Составить программу определения наибольшего собственного значения методом итераций, У(0) – произвольный вектор, можно взять его единичным.

Таблица 7 – Варианты заданий

Номер задания	Матрица напряжений				Точность *eps*	Входной файл	Выходной файл
	A_{i1}	A_{i22}	A_{i3}	A_{i4}			
3.7.1.	10,1 5 6 8	5 20,1 4 3	6 4 30 20	1 0,01 0,2 0,1	0,001	Создается редактором на диске	Вывод на терминал
3.7.2.	2,4 1 1,4	1 2,9 1,4	1,4 1,4 3,4	– – –	0,0001	Создается редактором на диске	Вывод на печать
3.7.3.	1,5 0,6 0,7	0,6 1,5 0,3	0,7 0,3 1,5	– – –	0,00001	Создается программно на диске	Вывод на терминал
3.7.4.	2,8 1,1 1,8	1,1 3,3 1,8	1,8 1,8 3,8	– – –	0,001	Ввод с терминала	Вывод на печать

Результаты расчета свести в таблицу:

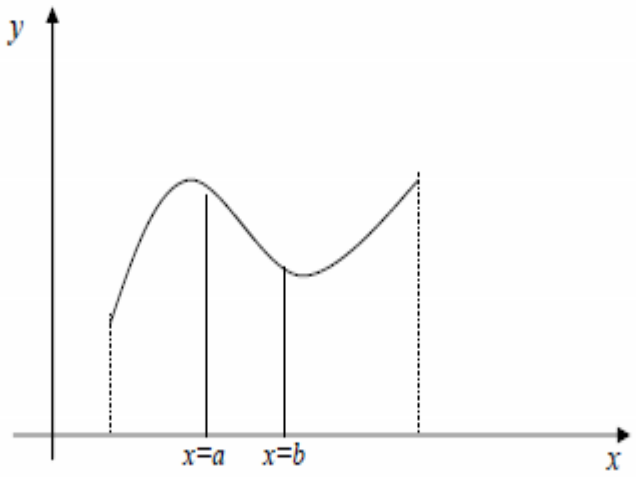
РАСЧЕТ СОБСТВЕННЫХ ЗНАЧЕНИЙ И СОБСТВЕННЫХ ВЕКТОРОВ МАТРИЦЫ НАПРЯЖЕНИЙ				
Собственное значение	Собственный вектор			
	A_1	A_2	A_3	A_4

3.8 Вычисление определенного интеграла

Определенным интегралом $\int_a^b f(x)dx$ функции $f(x)$ на отрезке [a,b] называется

$$\lim_{\substack{i=0 \\ \max \Delta x_i \to 0 \\ i}}^{n-1} \sum f(a_i) \cdot \Delta x_i \quad (1)$$

Величина определенного интеграла составляет площадь криволинейной трапеции, ограниченной графиком функции f(x), осью абсцисс и двумя прямыми x=a и x=b.



Разобьем отрезок [a,b] на n равных частей

$$h = {(b-a)}/{n} \quad (2)$$

и построим элементарные трапеции.

Общая площадь равна сумме элементарных трапеций и приближенно равна интегралу.

$$S = h((y_0 + y_1)/2 + (y_1 + y_2)/2 + \ldots + (y_{n-1} + y_n)/2) = \\ = ((b-a)/2n \cdot ((y_0 + 2y_1 + 2y_2 + \ldots + 2y_{n-1} + y_n)). \quad (3)$$

В методе Симпсона приближенное значение интеграла вычисляется:

$$S = (b-a)/b_n \cdot ((y_0 + 4y_1 + 2y_2 + \ldots + 4y_{n-1} + y_n). \quad (4)$$

Методы рассмотрены в [2], а так же в [3], [4], [5].

Составить программу вычисления интеграла, n принять равным 30. вычисление проводится с точностью eps, число n увеличивается до 2n и вновь вычисляется S_2, процесс счета заканчивается, когда

$$(S_k - S_{k+1}) < eps. \quad (5)$$

Таблица 8 – Варианты заданий

Номер задания	Подынтегральная функция	Точность	*a*	*b*	Входной файл	Выходной файл
3.8.1 Метод Симпсона	$\frac{1}{\sqrt{2x^2 + 0.3}}$	0,0001	0,7	1,3	Создается редактором на диске	Вывод на печать
3.8.2 Метод трапеций	$(x/2+1)\sin(x/2)$	0,001	1,2	2,8	Создается редактором на диске	Вывод на терминал
3.8.3 Метод трапеций	$\cos(x)$	0,00001	0	N/2	Ввод с терминала	Вывод на печать
3.8.4	$\sqrt{1 - \frac{1}{4}\sin^2 x}$	0,01	0	N/2	Создается программой на диске	Вывод на печать

Результаты расчета оформить в виде таблицы

РАСЧЕТ ИНТЕГРАЛА ПРИБЛИЖЕННЫМ МЕТОДОМ_____		
Значение интеграла	Кол-во итераций	Точность

Нина Александровна Ларина

ПРОГРАММИРОВАНИЕ
Часть II
(язык программирования Си в задачах)

Методическое пособие для студентов направления
230100 «Информатика и вычислительная техника»
дневной формы обучения

Редактор Е.Ф. Изотова

Подписано к печати 16.12.13. Формат 60х84 1/16.
Усл. печ. л. 7,19. Тираж 40 экз. Заказ № 131238. Рег. № 90.

Отпечатано в РИО Рубцовского индустриального института.
658207. Рубцовск, ул. Тракторная 2/6.