МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ В.Ф. УТКИНА»

КАФЕДРА

«ВЫЧИСЛИТЕЛЬНАЯ И ПРИКЛАДНАЯ МАТЕМАТИКА»

МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ «НИЗКО-УРОВНЕВОЕ ПРОГРАММИРОВАНИЕ»

Направление подготовки

09.03.04 «Программная инженерия»

Направленность (профиль) подготовки

Программная инженерия

Квалификация выпускника – бакалавр

Форма обучения – очная

Рязань

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Рекомендуется следующим образом организовать время, необходимое для изучения дисциплины:

Для освоения лекционного материала следует: изучить конспект лекции в тот же день, после лекции: 10 – 15 минут, повторно прочитать конспект лекции за день перед следующей лекцией: 10 – 15 минут. Также следует изучить теоретический лекционный материал по рекомендуемому учебнику/ учебному пособию: 1 час в неделю.

Следует максимально использовать лекционное время для изучения дисциплины, понимания лекционного материала и написания конспекта лекций. В процессе лекционного занятия студент должен уметь выделять важные моменты и основные положения. При написании конспекта лекций следует придерживаться следующих правил и рекомендаций.

1. При ведении конспекта рекомендуется структурировать материал по разделам, главам, темам. Вести нумерацию формул. Выделять по каждой теме постановку задачи, основные положения, выводы. Кратко записывать те пояснения лектора, которые показались особенно важными. Это позволит при подготовке к сдаче зачёта и экзамена не запутаться в структуре лекционного материала.
2. Лекционный материал следует записывать в конспект лишь после того, как излагаемый лектором тезис будет вами дослушан до конца и понят.
3. При конспектировании следует отмечать непонятные, на данном этапе, положения, доказательства и пр.
4. Рекомендуется по каждой теме выразить свое мнение, комментарий, вывод.

Подготовка к практическим занятиям:

Практические занятия по дисциплине существенно дополняют лекции. В процессе анализа теоретических положений и решения практических задач студенты расширяют и углубляют свои знания, полученные из лекционного курса и учебников, приобретают умение применять общие закономерности к конкретным случаям. В процессе решения задач развивается логическое мышление и вырабатываются навыки вычислений, работы со справочной литературой. Практические занятия способствуют закреплению знаний и практических навыков, формированию конструктивного стиля мышления, расширению кругозора.

При подготовке к практическому занятию необходимо внимательно ознакомиться с соответствующим теоретическим материалом по конспекту лекций и рекомендуемому учебнику, затем изучить конспект или материалы предыдущего практического занятия и выполнить заданное расчетное задание: 1 – 2 часа в неделю.

Следует максимально использовать аудиторное время практических занятий. В процессе занятия студент должен активно участвовать в дискуссиях, обсуждениях и решениях практических задач и вести конспект практических занятий отдельно от конспекта лекций.

Дополнительно в часы самостоятельной работы студенты могут повторно решить задачи, с которыми они плохо освоились во время аудиторных занятий, и обязательно те задачи, которые не получились дома при предыдущей подготовке к практическим занятиям.

Подготовка к лабораторным работам.

Перед началом проведения лабораторной работы необходимо ознакомится с методическими указаниями к данной лабораторной работе, внимательно ознакомиться с заданием и желательно заранее выполнить подготовку проекта в используемой инструментальной среде, чтобы время лабораторного занятия использовать для исправления ошибок, модификации проекта и защиты данной работы.

Выполнение каждой из запланированных работ заканчивается предоставлением отчета. Требования к форме и содержанию отчета приведены в методических указаниях к лабораторным работам или определяются преподавателем на первом занятии. Отчет по лабораторной работе студент должен начать оформлять еще на этапе подготовки к ее выполнению. Допускаясь к лабораторной работе, каждый студент должен представить преподавателю «заготовку» отчета, содержащую: оформленный титульный лист или название и номер работы при ведении общего конспекта, цель работы, задание, проект решения, полученные результаты, выводы.

Изучение методических указаний к лабораторной работе – 2 часа перед выполнением лабораторной работы и в ходе разработки проекта и 2 часа для оформления отчета, отладки проекта и подготовки к сдаче работы.

После выполнения лабораторной работы необходимо согласовать полученные результаты с преподавателем. Важным этапом является защита лабораторной работы. В процессе защиты студент отвечает на вопросы преподавателя, касающиеся теоретического материала, относящегося к данной работе, и проекта, реализующего его задание, комментирует полученные в ходе работы

результаты. При подготовке к защите лабораторной работы рекомендуется ознакомиться со списком вопросов по изучаемой теме и попытаться самостоятельно на них ответить, используя конспект лекций и рекомендуемую литературу. Кроме чтения учебной литературы рекомендуется активно использовать информационные ресурсы сети Интернет по изучаемой теме.

Подготовка к сдаче экзамена или зачета:

Экзамен/зачет – форма промежуточной проверки знаний, умений, навыков, степени освоения дисциплины. Главная задача экзамена/зачета состоит в том, чтобы у студента по окончанию изучения данной дисциплины сформировались определенное представление об общем содержании дисциплины, определенные теоретические знания и практические навыки, определенный кругозор. Готовясь к экзамену/зачету, студент приводит в систему знания, полученные на лекциях, на практических и лабораторных занятиях, разбирается в том, что осталось непонятным, и тогда изучаемая им дисциплина может быть воспринята в полном объеме с присущей ей строгостью и логичностью, ее практической направленностью.

Экзамены/зачеты дают возможность преподавателю определить теоретические знания студента и его практические навыки при решении определенных прикладных задач. Оцениваются: понимание и степень усвоения теоретического материала; степень знакомства с основной и дополнительно литературой, а также с современными публикациями; умение применить теорию к практике, решать определенные практические задачи данной предметной области, правильно проводить расчеты и т. д.; знакомство с историей данной науки; логика, структура и стиль ответа, умение защищать выдвигаемые положения.

Значение экзаменов/зачетов не ограничивается проверкой знаний, являясь естественным завершением обучения студента по данной дисциплине, они способствуют обобщению и закреплению знаний и умений, приведению их в стройную систему, а также устранению возникших в процессе обучения пробелов.

Подготовка к экзамену/зачету – это тщательное изучение и систематизация учебного материала, осмысление и запоминание теоретических положений, формулировок, формул, установление и осмысление внутрипредметных связей между различными темами и разделами дисциплины, закрепление теоретических знаний путем решения определенных задач.

Перед экзаменом назначается консультация, ее цель – дать ответы на вопросы, возникшие в ходе самостоятельной подготовки студента, студент имеет

возможность получить ответ на все неясные ему вопросы, кроме того, преподаватель будет отвечать на вопросы других студентов, что будет способствовать повторению и закреплению знаний всех присутствующих. Преподаватель на консультации, как правило, обращает внимание на те разделы, по которым на предыдущих экзаменах ответы были неудовлетворительными, а также фиксирует внимание на наиболее трудных разделах курса.

На непосредственную подготовку к экзамену обычно дается 3 – 5 дней. Этого времени достаточно для углубления, расширения и систематизации знаний, полученных в ходе обучения, на устранение пробелов в знании отдельных вопросов, для определения объема ответов на каждый из вопросов рабочей программы дисциплины.

Планируйте подготовку к зачету/экзамену, учитывая сразу несколько факторов: неоднородность в сложности учебного материала и степени его проработки в ходе обучения, свои индивидуальные способности. Рекомендуется делать перерывы в занятиях через каждые 50-60 минут на 10 минут. После 3-4 часов занятий следует сделать часовой перерыв. Чрезмерное утомление приведет к снижению тонуса интеллектуальной деятельности. Целесообразно разделять весь рабочий день на три рабочих периода – с утра до обеда, с обеда до ужина и с ужина до сна. Каждый рабочий период дня должен заканчиваться отдыхом не менее 1 часа. Работая в сессионном режиме, студент имеет возможность увеличить время занятий с 10 (как требовалось в семестре) до 12 часов в сутки.

Подготовку к экзаменам или зачетам следует начинать с общего планирования своей деятельности. С определения объема материала, подлежащего проработке, необходимо внимательно сверить свои конспекты с программой дисциплины, чтобы убедиться, все ли разделы отражены в лекциях, отсутствующие темы изучить по учебнику. Второй этап предусматривает системное изучение материала по данному предмету с обязательной записью всех выкладок, выводов, формул. На третьем этапе – этапе закрепления – полезно чередовать углубленное повторение особенно сложных вопросов с беглым повторением всего материала.

Рекомендации по работе с литературой:

Теоретический материал курса становится более понятным, когда дополнительно к прослушиванию лекции и изучению конспекта изучаются и книги по данному предмету. Литературу по дисциплине рекомендуется читать как в бумажном, так и в электронном виде (если отсутствует бумажный

аналог). Полезно использовать несколько учебников и пособий по дисциплине. Рекомендуется после изучения очередного параграфа ответить на несколько вопросов по данной теме. Кроме того, полезно мысленно задать себе следующие вопросы (и попробовать ответить на них): «о чем этот параграф?», «какие новые понятия введены, каков их смысл?», «зачем мне это нужно по специальности?».

Рекомендуется самостоятельно изучать материал, который еще не прочитан на лекции и не применялся на лабораторном или практическом занятии, тогда занятия будут гораздо понятнее. В течение недели рекомендуется выбрать время (1 час) для работы с литературой.

Лабораторная работа №1
Знакомство с языком С.
Ассемблер и ассемблерные вставки.

Цель работы:

Изучить базовые ассемблерные арифметические операции над целыми числами, а также способ взаимодействия встраивания кода ассемблера в программу на **C**.

Задание:

В соответствии с вариантом, необходимо рассчитать и вывести на экран выражение и его значение, рассчитанное с помощью ассемблерной вставки. Переменные A, B, C, D, E, F являются целочисленными и вводятся пользователем с клавиатуры.

Варианты заданий:

1. D + (A + F) - E * B * (D + C / D)
2. F / (A * B + C) – (D * C + E) / E
3. B + (A – D) / F - E * D / (C – F)
4. D / (B + C – F) + A * (C / D – E)
5. D * A / (D + A) - F / (B – E) + C
6. D / (C + B + E) + (F – B) * A / C
7. (C + F) / D – (B * E + F) * A - A
8. B * (C * E – D) / C + A * (F / D)
9. (E – A) * C - B * (C – F) - D / E

10. (B - F + E) * A + (A - D / C) * E
11. B * D / (F + D) – (B / E – A) * C
12. (D - E – B) / B * A + F * (C – D)
13. D / C * (F + D) + B * (F – A / E)
14. (E + C) * (F / A – D) * A + B / D
15. C * (A – D) - F / (B + E) + C / A
16. E * (D + F) - C / B - A * (E – C)
17. A * B / (E + F) – (C + D) * D / E
18. (D – B) / (C + A) * D - A + E - F
19. D * (E + A / C) - F * (B / F + A)
20. C / (B - E * C) + A / (F + B * D)
21. A + A / (F – E) – (B * E + D) / C
22. (C + B) / C * F + (E – D) / F * A
23. B + (C - C / F) / A + E * (E – D)
24. B * (E * F + A) - C * (F - D / E)
25. A * (C * F + D) – (D – C) / B - E
26. D * E * (C / B – C) – (E + F) / A
27. E * F / (F – C) + B * D / (A + B)
28. B / (F + D * A) - C * F / (B + E)
29. (F + B) * C - C + E / (B - D – A)
30. F + (F – B) * A * E + D / (A + C)
31. (B + E – A) * E * F - F / (C – D)
32. C + D * F / (F – E) + (A – B) * F

Лабораторная работа №2
Управляющие структуры языка С.

Цель работы:

Изучение основных управляющих структур языка **С**. Проектирование программы по принципу нисходящего проектирования.

Задание:

Табулируйте функцию двух аргументов и найдите те значения аргументов, при которых функция принимает *максимальное* и *минимальное* значение в заданном диапазоне.

В заданиях функция **f** табулируется по аргументам **x [x0 (xh) xn]** и **y[y0 (yh) yn]**, а параметры **a, nm1, nm2** вводятся пользователем произвольно, причём **nm1, nm2 [2,6]**

При проектировании алгоритма программы применить нисходящее проектирование.

Варианты заданий:

Вариант 1.

$$\sqrt{nf(x,y)} = \prod_{nm=2}^{nm1} x_n + (-1)$$

y_n , *если* x a;

n 0

$(\sin^2 x + \cos^3(yx^{n+1}+1))$ *при других* x;

$n=1$

Вариант 2.

$$f(x,y) = \begin{cases} \dfrac{x + y^{nm1}(2x+1)^n}{\sqrt{x + 4}\,\sum_{nmn2=1}} & \text{, если } x \; a; \\ \dfrac{}{2n(+yxy+1)_n \sqrt{x+1}} & \text{при других } x; \end{cases}$$

$$\left(1 + \sum_{n=1}^{} y_{n+1}\right)$$

Вариант 3.

$$\sum_{n=0}^{} 2y_{n+1} + xn \text{ , если } x \; a; \quad nm1 \quad x_n$$

$$f(x,y) = \begin{cases} \\ \end{cases}$$

Вариант 4.

$$\begin{cases} \\ \end{cases}$$

n , *если* $x + y$ a;

$$\begin{cases} \\ \end{cases}$$

если $x + y$ a;

Вариант 5.

$$\begin{cases} \\ \end{cases}$$

если x a;

$$f(x,y) = \begin{cases} \\ \end{cases}$$

если x a;

Вариант 6.

$$= \begin{cases} \\ \end{cases}$$

Вариант 7.

$$\begin{cases} \\ \end{cases}$$

если x ay;

$f(x,y) = \square\square\square\frac{yx+1}{3}\ _{nm}\square\sqrt{\ }_{n=12}x8+ny+_21_n,\quad если\quad ay\square x;$

Вариант 8.

$$\square_{nm1}x+2y_{n+1}$$

$$f(x,y) = \square\square\square\square_{nmn=02}(2y+1)_n, если\ y\square a;$$

$$\square\square\square_{n=0}\ln(yx)\square(\sin(x-a)+n), если\quad y\square a;$$

Вариант 9.

$$_{nmn=11}{}^{n})+x+1\ _{nm}{}^{\square}{}_{=2}7x2\sqrt[\gamma]{n+y_ny^{-1}};$$

$$f(x,y) = \square(1+y\ln x\ 3y_{n1}$$

Вариант 10.

$$\square y_{nm1}\square x_n\square$$

$$\square f(x,y) = \square\square x+1\ _{n=n1+1}\square yn\square \quad \_\_\square\square y+\_\_\square\square, \quad если\quad x\square a;$$

$$_{nm2}$$

$$\square\ \frac{y}{\sqrt{\ }_{n=0}},\quad x\quad a;$$

$$\square\square\square\quad n+1\quad если\quad \square f(x,y)$$

Вариант 11.

$$x_2\quad y_{nm1}\square\quad \frac{x_n}{\_\_\_}\square\ \frac{x_3\quad 2y_{nm2}y_n}{\_\_\_\_\_}$$

$$=\_\_\_\_\_+\ \square_{=}\square\square 1+n+y\square\square\square+\quad +4\ \square_{n=1}n$$

$$;\ x\quad _{n1}\square$$

Вариант 12.

$$f(x, y) = \overline{x_2 + 23..58\square\square 1010\overline{\sqrt{-34}\square_{nmn=11}\overline{(yx3n++\overline{24})_n}}} + xx++12_{nm}\square_{n=12}(x+n3y)_n;\ x+$$

Вариант 13.

$$\sum_{k=0}^{nm1}\prod_{n=1}^{nm2}\frac{\sqrt{}}{n}$$

Вариант 14.

$$\square a_{nm1}x_n - yn$$

, $a \quad x \quad y;$

$\sin xy \quad {}^{n} \quad 5y^3));\quad f(x, y) = \square(kx+\square(k$

$++3x+$

$$\square 3\square_{n=1}\sqrt{4a_2 = x} \quad \textit{если} \quad \square +$$

$$\square\ f(x, y) = \square\square a\ +1_{nm}\square_2\square 1+ (ay-1)_n\square\square,$$

если $a\overline{\square x} + y;$

$$\square\square\ a_{n=1}\square\square \quad \overline{\sqrt{xn}}\ \square$$

Вариант 15.

$$f(x,y)=\begin{cases}\prod_{nm1}\sqrt{yn+x} & \text{если}\\ \sum_{n=1}nx_{n+1} & \\ +1, & x\ ay;\\ \prod_{nm2}x-y & \\ , & x\quad ay;\end{cases}\qquad \sum_{n=0}\quad n+y\quad \text{если}\quad \sqrt{\quad}$$

Вариант 16.

$$f(x,y)=\begin{cases}x_{nm1}\prod & yx_n\quad x_{n+1}\\ 1 & x+y_{nm2}\prod\\ y & \end{cases}\qquad =2^{=}+n+x_2+1\Big)_{n=1}+2n-5$$

$$n\,1$$

Вариант 17.

$$f(x,y)=\begin{cases}\sum_{nm1}\cos^3\left(yx^{n+1}+1\right), & \text{если}\quad ||\quad x\ a;\\ \sum_{nnm=02}\sqrt{\frac{x^n+n}{yn}}) & \\ \sum_{n=1}(-1)^n(\sin^2x+ & \text{при}\quad \text{других}\quad x;\end{cases}$$

Вариант 18.

$$f(x,y)=\begin{cases}\underline{x+y}^{nm1}\quad \underline{(yx+1)^n} & ||\\ x_2+4 & {}_{n=1}1+n+4\quad ,\quad \text{если}\quad x\ a;\\ \prod_{nm^2}\left(2x_{\ n}\right. & \\ \sqrt{\quad}\ x+1\sqrt{}=xy+2n^{+}+13)y & \text{придругихx};\\ y_{\ n\,1} & \end{cases}$$

Вариант 19.

$\square_{nm1}\, x_n$

$\square^{\square}_{=} 3n$,*если* $x \square a$;

$\square_{n\ 0}\ f\ (x,\ y)\ =\ \square\square\square_{nm}\square_{=2}$

$2y3_n x_{+1} + + yxn$, *если* $x \square a$;

$\square_{n\,0}$

Вариант 20.

$\square_{nm1}\square\ y_n\ \square^n f\,(x, y) = \square\square^{\square}_{=}\ \square\square n$

$+\ 2\square\square\square$, $\sqrt{\ }$ *если* $x + y \square a$;

$\square_{n\ 1}\square$

$\square\square_{nm}\square_{2}\ \sin x + x_n + _{+1} y$, *если* $x + y \square a$;

$\square_{n=0}$ $2y$ $+n$

Вариант 21.

$\square x_{\ nm1}\, n + (3x + y)_{n+2}$

$\square\square 2\ \square_{n=1}\ yn + \sqrt{3}\ x$, $f\,(x, y) = \square^{\square}_{nm}\square_{=2}\ nx_n + 1$,

если $\sqrt{\ }$

$\square\square_{n\ 1}$

Вариант 22.

$\underline{yx+1}_{\ nm1}\square$ x_{2n} $\square$

$f\,(x, y)$ $= 2\ ^{\square\sqrt{\ }}_{=}\ \square\square 1 + 2n + 3y\ \square\square\square +$

$.5_{\ n\ 1}\square$

Вариант 23.

$\square\ yx + 1_{\ nm1}\, x_{n+1}$

$\square^{\square}\ 3$

$\square_{n=0}\ 8n + 1$, *если* f

$(x, y) = \square_{nm2}\, x + y_{2n}$

если $x \square a$;

x

$\square$

a

; $_{nm2}$ x_n

$\sum_{n=1} yn + 2$, $x \neq ay$;

$\sqrt{\sum_{n=1} yn + 3}$, *если* $ay \neq x$;

Вариант 24.

$$\sum_{nm1} \frac{(2y+1)_n}{2y_{n+1}}, \quad \text{если } y \neq a;$$

$$f(x,y) = \sum_{nm_{n0}^2} (\sin(x-a) + \sqrt{n}) \sum_{n0} \ln(x) \quad x + y_{2n}, \quad \text{если } y \neq a;$$

Вариант 25.

$$f(x,y) = \frac{x3+y1}{\sum_{nm=1} \frac{1}{1 + {}_{n1}\sum \sqrt{72xn++yy_{n-1}} + {}_{nm}\sum_{n=12} y\ln x_n}};$$

Вариант 26.

$$\square_{nm1}\square\ y_{n+1}\square$$

$$\square\square\ \square_{n=1}\square\square\square n+1\square\square\square\ ,$$

$f(x,y)=\square\ y_{nm2}\square$ *если* $x\square a$;

$$\sqrt{\ \ \square\ \ \ \ \square_{=}\square\square\ y+yn\ \square\square,\quad \text{если}\quad x\square a;}$$
$$\square\square x+1_{n\,0}\square\square$$

Вариант 27.

$$f(x,y)=\frac{x_3+2y_{nm1}\square\quad y_n\square}{\sqrt{\ }}\ \frac{\sqrt{x_2+y_{nm2}}\quad x_n}{\square\ \sqrt{\ }}$$
$$\square_{=}\square\square 1+n\ \square\square+x_{n=1}\quad n+y\ ;$$
$$n\,1\square$$

Вариант 28.

$$f(x,y)=\frac{\sqrt{\ }}{2\quad 3x.8+\square 210^{-4}\square_{nmn=1}}\ \frac{\sqrt{\ }}{m+n\,4+}\quad x^{\ }$$
$$+x2+.51\square 10^{-3\,nm=2}(x+y)_n\,;\ x+\square(yx+2)_n$$
$$n\,1$$

Вариант 29.

$$f(x,y)=\square_{k=0}^{nm1}(kx+\square_{n=1}^{nm2}\sqrt{\frac{k+\sin xy}{(\quad 2n\quad +\quad 3k\quad)}}\ \frac{3x_n+5y_{n-3}}{\sqrt{\ }});$$

$x_n\square$

Вариант 30.

□□*a* −4 *n*=1□□

□*a* *nm*1□ (*ay*−1)*n* □ *xn* □

— *a* □ *x* + *y*;

□3 *n* 1□ $\frac{4a -}{\sqrt{}}$ *a* □ *x* + *y*;

f (*x*, *y*) $\frac{=□□^{□}a}{\sqrt{}}$

+□ =1□*nm*□*n*²+□1+ *x*3*n*

¯*xyn*□□,□□, *еслиесли*

Вариант 31. □*nm*1

$$\begin{cases} 2n + x \\ □□ \sqrt{\sqrt{}}_{n=1} \sqrt{n +} y \text{ , если } x □ ay; \end{cases}$$

$$f(x, y) = □_{nm2} x - 3y$$

$$\begin{cases} □□_{=} nx_{n+1} + 1, & \text{если } x □ ay; \\ □ \\ □_{n\,0} \end{cases}$$

Вариант 32.

$$f(x, y) = 2 \quad - \; □\sqrt{}_{n=1} □□□1 + \; \frac{}{2n - + 5} □□□ + \; \frac{\sqrt{}}{}$$

$$x_2 + + \frac{y}{1nm} □ \; {}_{n=12}□□□□ \; y + {}^{yx}n_{\,n} □□□□ \; x_{\,nm1}□$$

$$x_{n\,1}□ \quad x$$

Лабораторная работа №3
Массивы и функции языка С.

Цель работы:

Изучить принципы работы со статическими массивами и функциями языка С.

Задание:

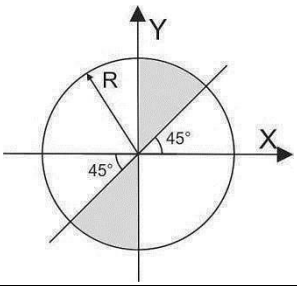
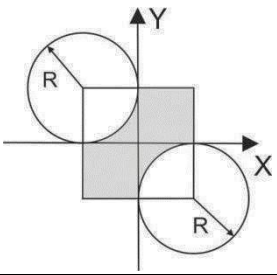
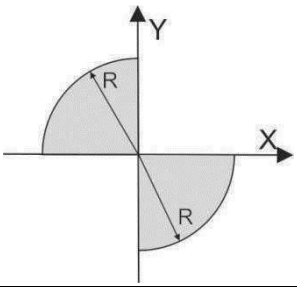
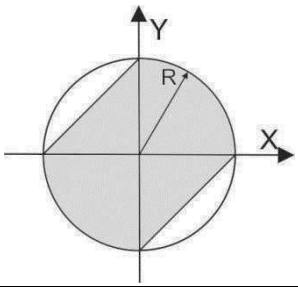
Дан массив координат, необходимо определить какие из этих координат попадают в заданную закрашенную область, а какие не попадают, и вывести эту информацию на экран. Массив координат задаётся двумерным массивом, где первый индекс означает номер координаты, а второй номер компоненты координаты (**x** – 0, **y** - 1). Массив задаётся по выбору пользователя либо через непосредственный ввод координат с клавиатуры, либо случайным образом в заданном пользователем диапазоне.

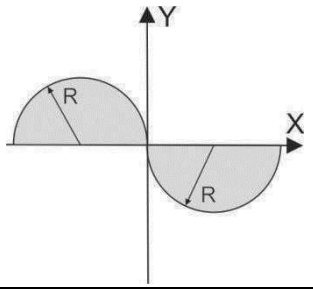
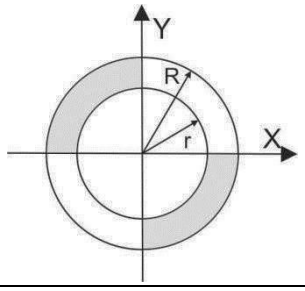
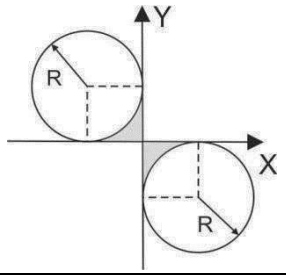
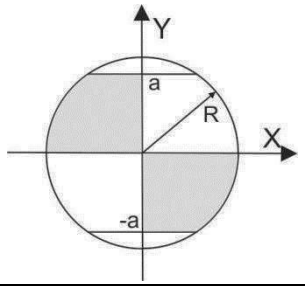
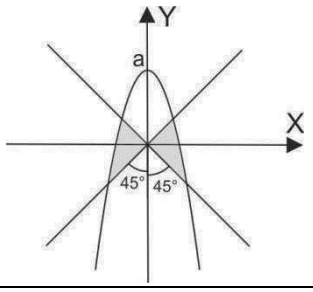
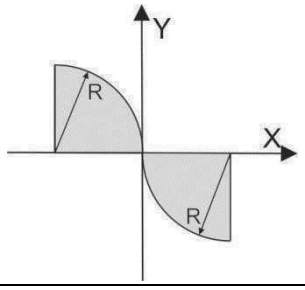
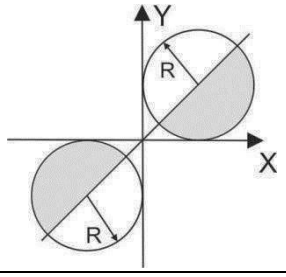
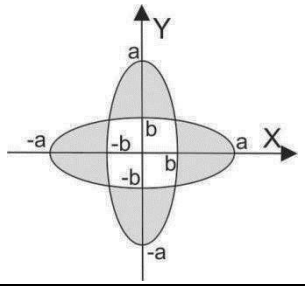
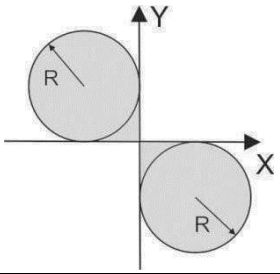
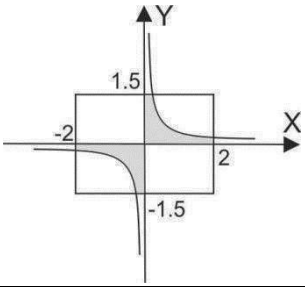
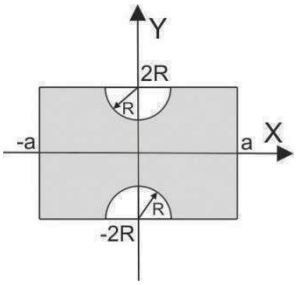
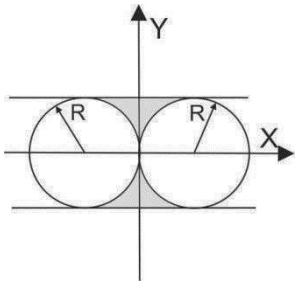
В программе должно быть минимум четыре отдельных функции:

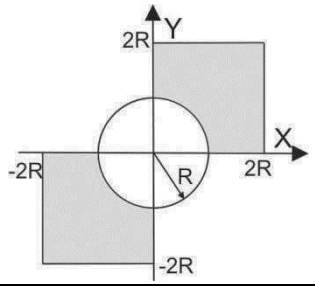
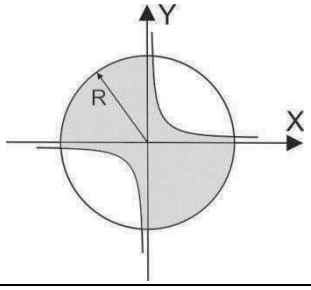
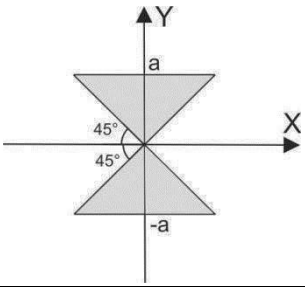
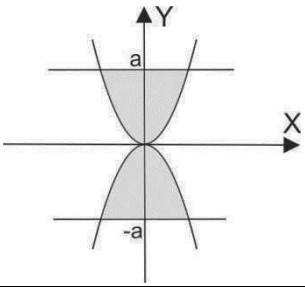
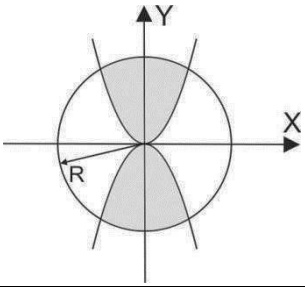
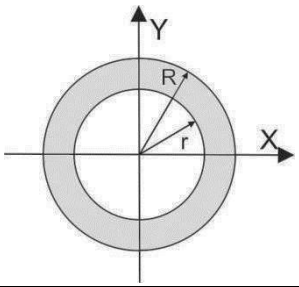
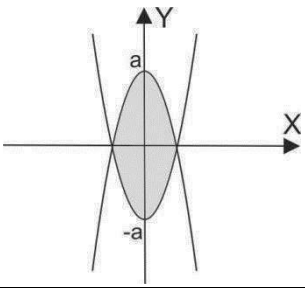
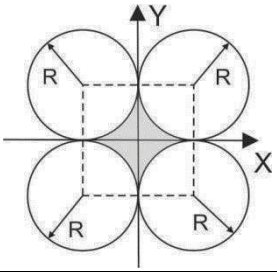
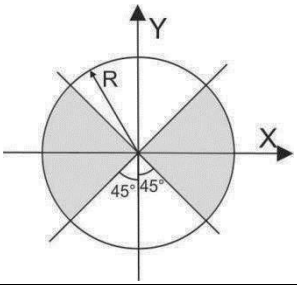
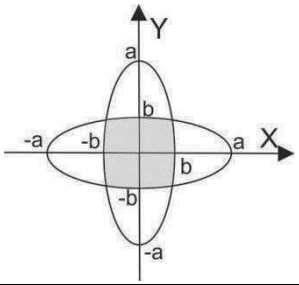
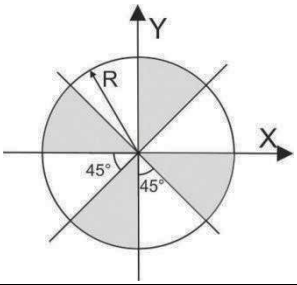
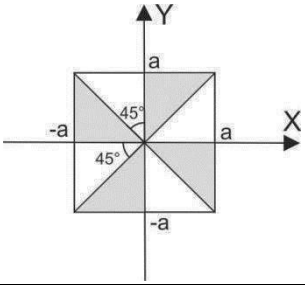
1. Задание массива координат через непосредственный ввод координат с клавиатуры.
2. Задание массива координат случайным образом в заданном диапазоне.
3. Определение попадание точки в заданную область.
4. Вывод попавших и не попавших координат в область.

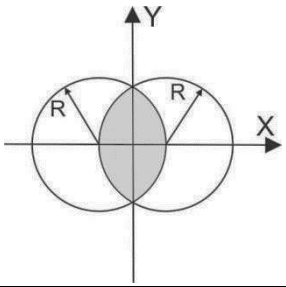
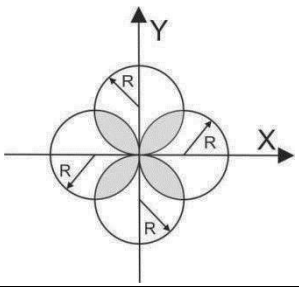
Параметры **R, a** и **b**, задающие область, а также количество координат задаются пользователем.

Варианты заданий:

№ Вар	**Область**	**№ Вар**	**Область**
			
			

	 Y X R R		 Y X R r
	 Y X R R		 Y X a R -a
	 Y X a 45° 45°		 Y X R R
11	 Y X R R		 Y X a -a -b b a b -b -a
13	 Y X R R		 Y X 1.5 -2 2 -1.5
15	 Y X 2R R -a a R -2R		 Y X R R

17			
19			
21			
23			
25			
27			

29			

Лабораторная работа №4 Указатели.

Цель работы:

Изучение принципов работы указателей в языке С, способов передачи массивов в функции через указатели, а также применения в программе указателей на функции.

Задание:

Преобразовать лабораторную работу №3 таким образом, чтобы работа с массивами, и их передача в качестве параметров в функции велась только через арифметику указателей. Выбор функции заполнения массива необходимо осуществлять через указатель на функцию.

Лабораторная работа №5
Динамическая память.

Цель работы:

Изучение принципов работы с динамическими массивами в языке С. Реализация операций «Удаление», «Добавление», «Перестановка», «Поиск» для динамических массивов.

Задание:

В каждом варианте задания требуется создать **целочисленный** динамический двумерный массив, который может становиться «неправильной формы», т.е. каждая строка которого может быть своей длины, причём индексация строк начинается с 1, а в нулевом элементе хранится общее количество элементов данной строки. В начале массив

генерируется заданного пользователем размера A x B с элементами заданными случайным образом из заданного пользователем диапазона, а все строки которого изначально будут одинаковой длины B. В последствии необходимо к каждой строке применить одну из 4-х функций по модификации одномерного динамического массива (по классу задач: удаление, добавление, перестановка, поиск), заданных ниже по-вариантно, и в результате удаления\вставок элементов каждая строка может изменять свой размер. Функции к строкам применяются последовательно: т.е. функция «удаление» применяется к 1 строке двумерного массива, функция «добавление» применяется ко 2 строке, функция «перестановка» применяется к 3 строке, функция «поиск» применяется к 4 строке, функция «удаление» применяется к 5 строке, функция «добавление» применяется к 6 строке и т.д. При добавлении элементов соответственно строки динамического массива должны расширяться, а при удалении элементов – уменьшаться. Весь текстовый ввод\вывод должен осуществляться в консоль исключительно из основной функции **main**.

Важно:

1) Функции работают с указателями на динамические **одномерные** массивы, т.е. строки двумерного массива мы передаём построчно для обработки в функции как **одномерные** массивы через указатели.

2) Функции работают с **целочисленными** динамическими одномерными массивами где индексы начинаются с 1, а в 0-вом элементе записан текущий размер динамического массива. При удалении\добавлении элементов функции должны записать новый размер массива в 0-вой элемент.

3) В функциях будет более удобно использовать арифметику указателей и относительные смещения.

4) При добавлении случайных элементов по заданию нужно использовать тот же диапазон генерации, который задал пользователь для генерации исходного двумерного массива.

5) Циклический сдвиг вправо или влево – это когда элементы из конца или начала массива переходят соответственно в его начало или конец.

Варианты заданий:

	Функции для работы с одномерным динамическим массивом

Вариант	Удаление	Добавление	Перестановка	Поиск
1	Максимальный элемент (включая его повторения)	К случайных элементов в в начало массива	Перевернуть массив	Занулить первый четный элемент
2	Минимальный элемент (включая его повторения)	К случайных элементов в в конец массива	Сдвинуть цикличеcки на M элементов вправо	Занулить последний четный элемент
3	Элемент с заданным номером K	К случайных элементов в в середину массива	Сдвинуть цикличеcки на M элементов влево	Занулить первый отрицательный элемент
4	N элементов, начиная с номера K	N случайных элементов, начиная с номера K	Сдвинуть цикличеcки только нулевые элементы на M элементов вправо	Занулить последний отрицательный элемент

	5	N элементов, перед элементом с номером K	N случайных элементов, перед элементом с номером K	Сдвинуть цикличеcки только нулевые элементы на M элементов влево	Занулить все элементы с заданным значением
	6	Все четные элементы	Случайный элемент с номером K	Четные элементы переставить в начало массива, нечетные - в конец	Занулить первый элемент с заданным значением
	7	Все нечетные элементы	По одному случайному элементу перед каждым нулевым элементом	Поменять местами минимальный и максимальный элементы (учитывая их повторения)	Занулить последний элемент с заданным значением

	8	Все нулевые элементы	По одному случайному элементу после каждого отрицательного элемента	Положительные элементы переставить в начало массива, отрицательные - в конец	Заменить все отрицательные элементы среднеарифметическим значением элементов массива

	9	Все элементы с четными индексами	Добавить в начало столько нулевых элементов, сколько по модулю значение максимального элемента массива	Переставить все нулевые элементы в начало массива	Заменить все элементы с заданным значением на среднее арифметическое всех элементов массива
	10	Все элементы с нечетными индексами	Добавить в конец столько нулевых элементов, сколько по модулю значение максимального элемента массива	Переставить все нулевые элементы в конец массива	Занулить элементы равные среднему арифметическому элементов массива
	11	Все элементы больше среднего арифметического элементов массива	К случайных элементов в начало массива	Перевернуть массив	Занулить последний элемент с заданным значением

	12	Все элементы меньше среднего арифметического элементов массива	К случайных элементов в конец массива	Сдвинуть цикличеcки на М элементов вправо	Заменить все отрицательные элементы среднеарифметическим значением элементов массива
	13	Максимальный элемент (включая его повторения)	К случайных элементов в середину массива	Сдвинуть цикличеcки на М элементов влево	Заменить все элементы с заданным значением на среднее арифметическое всех элементов массива
	14	Минимальный элемент (включая его повторения)	N случайных элементов, начиная с номера К	Сдвинуть цикличеcки только нулевые элементы на М элементов вправо	Занулить элементы равные среднему арифметическому элементов массива
	15	Элемент с заданным номером К	N случайных элементов, перед элементом с номером К	Сдвинуть цикличеcки только нулевые элементы на М элементов влево	Занулить первый четный элемент
	16	N элементов, начиная с номера К	Случайный элемент с номером К	Четные элементы переставить в начало массива, нечетные	Занулить последний четный элемент

			- в конец	
	17 N элементов, перед элементом с номером K	По одному случайному элементу перед каждым нулевым элементом	Поменять местами минимальный и максимальный элементы (учитывая их повторения)	Занулить первый отрицательный элемент
	18 Все четные элементы	По одному случайному элементу после каждого отрицательного элемента	Положительные элементы переставить в начало массива, отрицательные - в конец	Занулить последний отрицательный элемент
	19 Все нечетные элементы	Добавить в начало столько нулевых элементов, сколько по модулю значение максимального элемента массива	Переставить все нулевые элементы в начало массива	Занулить все элементы с заданным значением

20	Все нулевые элементы	Добавить в конец столько нулевых элементов,	Переставить все нулевые элементы в конец массива	Занулить первый элемент с заданным значением

		сколько по модулю значение максимального элемента массива		
21	Все элементы с четными индексами	К случайных элементов в начало массива	Перевернуть массив	Занулить последний элемент с заданным значением
22	Все элементы с нечетными индексами	К случайных элементов в конец массива	Сдвинуть цикличеcки на M элементов вправо	Заменить все отрицательные элементы среднеарифметическим значением элементов массива
23	Все элементы больше среднего арифметического элементов массива	К случайных элементов в середину массива	Сдвинуть цикличеcки на M элементов влево	Заменить все элементы с заданным значением на среднее арифметическое всех элементов массива

	24	Все элементы меньше среднего арифметического элементов массива	N случайных элементов, начиная с номера K	Сдвинуть цикличеки только нулевые элементы на M элементов вправо	Занулить элементы равные среднему арифметическому элементов массива
	25	Максимальный элемент (включая его повторения)	N случайных элементов, перед элементом с номером K	Сдвинуть цикличеки только нулевые элементы на M элементов влево	Занулить все элементы с заданным значением
	26	Максимальный элемент (включая его повторения)	Случайный элемент с номером K	Четные элементы переставить в начало массива, нечетные - в конец	Занулить первый элемент с заданным значением
	27	Минимальный элемент (включая его повторения)	По одному случайному элементу перед каждым нулевым элементом	Поменять местами минимальный и максимальный элементы (учитывая их повторения)	Занулить последний элемент с заданным значением

	28	Элемент с заданным номером K	По одному случайному элементу после каждого отрицательного элемента	Положительные элементы переставить в начало массива, отрицательные - в конец	Заменить все отрицательные элементы среднеарифметическим значением элементов массива
	29	N элементов, начиная с номера K	Добавить в начало столько нулевых элементов, сколько по модулю значение максимального элемента массива	Переставить все нулевые элементы в начало массива	Занулить первый четный элемент
	30	N элементов, перед элементом с номером K	Добавить в конец столько нулевых элементов, сколько по модулю значение максимального элемента массива	Переставить все нулевые элементы в конец массива	Занулить последний четный элемент

	3 1	Все четные элементы	К случайных элементов в в середину массива	Перевернуть массив	Занулить первый отрицательный элемент
	3 2	Все нечетные элементы	N случайных элементов, начиная с номера К	Сдвинуть цикличеcки только нулевые элементы на М элементов влево	Занулить последний отрицательный элемент
	3 3	Все нулевые элементы	N случайных элементов, перед элементом с номером К	Четные элементы переставить в начало массива, нечетные - в конец	Занулить все элементы с заданным значением

Лабораторная работа №6 Строки и параметры запуска.

Цель работы:

Изучение принципов работы со строками в языке С. Использование параметров запуска для передачи данных в запускаемую программу.

Задание:

В каждом варианте задания необходимо создать программу, принимающую в качестве параметров запуска первым аргументом текст для обработки, вторым аргументом - команду обработки (одну из трёх), и после нее необходимые для работы команды дополнительные аргументы. Все команды обработки делятся на три вида: информация, создание, удаление (заданы ниже по-вариантно). Таким образом запуск программы должен иметь следующий формат:

Lab6.exe "Текст для обработки идёт первым аргументом." -info 5

Где второй аргумент задаёт команду, соответственно:

-info для команды «информация».

-create для команды «создание». **-delete**

для команды «удаление».

Третьим и далее аргументами идёт необходимый набор параметров для каждой из этих команд. В каждом варианте задания требуется создать минимум три функции реализующие соответствующие команды. При применении команды «информация» в консоль следует вывести искомое количество, при «создании» вывести созданный массив в консоль, а при «удалении» вывести в консоль модифицированный текст. При любом сравнении последовательностей не учитывать регистр букв. В команде «создание» использовать динамические массивы. Весь текстовый вывод в консоль должен осуществляться исключительно из основной функции **main.** Программа также должна правильно обрабатывать случай, когда аргументы запуска отсутствуют либо заданы неверно, и выводить текст ошибки, поясняющий что конкретно было сделано неправильно при задании параметров.

Важно: любой текст может состоять из *слов*, *чисел*, либо иных *последовательностей* символов. Также текст может быть разбит знаками препинания на предложения.

Слово – последовательность символов, состоящая из русских букв верхнего или нижнего регистра.

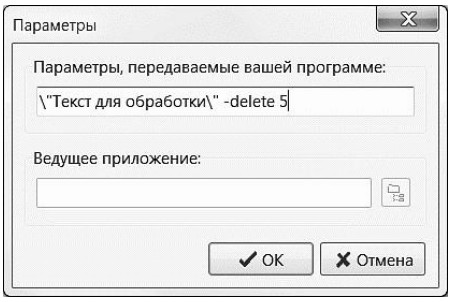
Число – последовательность символов, состоящая из цифр, которая также может иметь вначале символ знака “+” или “-”. Числа считаются только целыми. Дробные числа следует относить к иным последовательностям.

Предложение в тексте может заканчиваться на символы: точка “.”, восклицательный знак “!”, вопросительный знак “?” или конец строки ‘\0’.

Следует также учитывать, что слова, числа и другие последовательности могут разделяться не только пробелами и знаками конца предложения, но и символом запятая “,”.

Примечание о параметрах запуска: если вы запускаете программу с параметрами запуска через среду Dev-C++ (выполнить -> параметры…) то

текст в качестве первого аргумента следует взять в кавычки с обратным слешем:



Альтернативный вариант: для тестирования программы также можно создать в той же папке, что и сама программа, текстовый файл с расширением ***.bat** (т.е. пакетный или командный файл) и внести в него следующие содержание в любом текстовом редакторе:

```
Lab6.exe "Текст для обработки" -delete 5
Pause
```

Таким образом, запуская данный пакетный файл, можно передавать приложению требуемые параметры для запуска. В пакетном файле обратный слеш перед кавычками ставить **не нужно**. Команда **pause** нужна чтобы консоль вывода не закрывалась сразу после окончания программы.

Варианты заданий:

Вариан	**Команды**		
	Информация	**Создание**	**Удаление**
1	Функция, возвращающая общее количество слов в тексте.	Функция, создающая массив слов, содержащих больше гласных чем согласных.	Функция, которая удаляет каждое K-ое слово из текста.
2	Функция, возвращающая общее количество предложений в тексте.	Функция, создающая массив слов, заканчивающихся на гласную букву.	Функция, которая удаляет N-ое предложение из текста.

3	Функция, возвращающая среднее арифметическое всех длин слов в тексте.	Функция, создающая массив уникальных гласных букв, встречающихся в словах в K-ом предложении.	Функция, удаляющая все первые слова в предложениях в тексте.
4	Функция, возвращающая сумму всех чисел, встречающихся в тексте.	Функция, создающая массив слов, длинней K-букв.	Функция, удаляющая все последние числа в предложениях в тексте.
5	Функция, возвращающая максимальную длину слова, встречающуюся в тексте.	Функция, создающая массив чисел, меньше чем M.	Функция, которая удаляет каждое K-ое число из текста.
6	Функция, возвращающая общее количество последовательностей в тексте, не являющихся числами или словами.	Функция, создающая массив уникальных цифр, встречающихся в числах в K-ом предложении.	Функция, которая удаляет каждое предложение в тексте, если в нем есть числа.

7	Функция, возвращающая количество слов в тексте, в которых встречаются сдвоенные буквы.	Функция, создающая массив предложений, в которых более M слов.	Функция, удаляющая из текста все последовательности, не являющиеся ни словами, ни числами.
8	Функция, возвращающая общее количество чисел в тексте.	Функция, создающая массив из всех первых слов предложений.	Функция, которая удаляет все слова из текста, содержащие заданную последовательность букв.
9	Функция, возвращающая сколько раз встречается заданное слово (или часть слова) в тексте.	Функция, создающая массив из всех последних чисел предложений.	Функция, которая удаляет все слова из текста, длина которых больше K.
10	Функция, возвращающая минимальную длину слова, встречающуюся в тексте.	Функция, создающая массив из всех последовательностей, не являющихся числами или словами.	Функция, которая удаляет все числа из текста, сумма цифр которых меньше M.
11	Функция, возвращает количество гласных букв, встречающихся в словах текста.	Функция, создающая массив предложений, в которых более M чисел.	Функция, которая удаляет из текста все предложения, в которых встречаются числа.
12	Функция, возвращает количество последовательностей в тексте, в которых есть цифры.	Функция, создающая массив слов, содержащих заданную последовательность букв.	Функция, которая удаляет каждое K-ое слово из текста.

13	Функция, возвращает количество чётных чисел в тексте.	Функция, создающая массив чисел, сумма	Функция, которая удаляет N-ое предложение из текста.

		цифр которых больше K.	
14	Функция, возвращающая общее количество слов в тексте.	Функция, создающая массив последовательностей, в которых есть цифры.	Функция, удаляющая все первые слова в предложениях в тексте.
15	Функция, возвращающая общее количество предложений в тексте.	Функция, создающая массив слов, содержащих больше гласных чем согласных.	Функция, удаляющая все последние числа в предложениях в тексте.
16	Функция, возвращающая среднее арифметическое всех длин слов в тексте.	Функция, создающая массив слов, заканчивающихся на гласную букву.	Функция, которая удаляет каждое K-ое число из текста.
17	Функция, возвращающая сумму всех чисел, встречающихся в тексте.	Функция, создающая массив уникальных гласных букв, встречающихся в словах в K-ом предложении.	Функция, которая удаляет каждое предложение в тексте, если в нем есть числа.
18	Функция, возвращающая максимальную длину слова, встречающуюся в тексте.	Функция, создающая массив слов, длинней K-букв.	Функция, удаляющая из текста все последовательности, не являющиеся ни словами ни числами.

19	Функция, возвращающая общее количество последовательностей в тексте, не являющихся числами или словами.	Функция, создающая массив чисел, меньше чем M.	Функция, которая удаляет все слова из текста, содержащие заданную последовательность букв.
20	Функция, возвращающая количество слов в тексте, в которых	Функция, создающая массив уникальных цифр, встречающихся в	Функция, которая удаляет все слова из текста, длина которых больше K.

	встречаются сдвоенные буквы.	числах в K-ом предложении.	
21	Функция, возвращающая общее количество чисел в тексте.	Функция, создающая массив предложений, в которых более M слов.	Функция, которая удаляет все числа из текста, сумма цифр которых меньше M.
22	Функция, возвращающая сколько раз встречается заданное слово (или часть слова) в тексте.	Функция, создающая массив из всех первых слов предложений.	Функция, которая удаляет из текста все предложения, в которых встречаются числа.
23	Функция, возвращающая минимальную длину слова, встречающуюся в тексте.	Функция, создающая массив из всех последних чисел предложений.	Функция, которая удаляет каждое K-ое слово из текста.
24	Функция, возвращает количество гласных букв, встречающихся в словах текста.	Функция, создающая массив из всех последовательностей, не являющихся числами или словами.	Функция, которая удаляет N-ое предложение из текста.

25	Функция, возвращает количество последовательностей в тексте, в которых есть цифры.	Функция, создающая массив предложений, в которых более M чисел.	Функция, удаляющая все первые слова в предложениях в тексте.
26	Функция, возвращает количество чётных чисел в тексте.	Функция, создающая массив слов, содержащих заданную последовательность букв.	Функция, удаляющая все последние числа в предложениях в тексте.
27	Функция, возвращающая общее количество слов в тексте.	Функция, создающая массив чисел, сумма цифр которых больше K.	Функция, которая удаляет каждое K-ое число из текста.
28	Функция, возвращающая общее количество предложений в тексте.	Функция, создающая массив последовательностей, в которых есть цифры.	Функция, которая удаляет каждое предложение в тексте, если в нем есть числа.
29	Функция, возвращающая среднее арифметическое всех длин слов в тексте.	Функция, создающая массив слов, содержащих больше гласных чем согласных.	Функция, удаляющая из текста все последовательности, не являющиеся ни словами, ни числами.
30	Функция, возвращающая сумму всех чисел, встречающихся в тексте.	Функция, создающая массив слов, заканчивающихся на гласную букву.	Функция, которая удаляет все слова из текста, содержащие заданную последовательность букв.
31	Функция, возвращающая максимальную длину слова, встречающуюся в тексте.	Функция, создающая массив уникальных гласных букв, встречающихся в словах в K-ом предложении.	Функция, которая удаляет все слова из текста, длина которых больше K.

32	Функция, возвращающая общее количество последовательностей в тексте, не являющихся числами или словами.	Функция, создающая массив слов, длинней K-букв.	Функция, которая удаляет все числа из текста, сумма цифр которых меньше M.
33	Функция, возвращающая количество слов в тексте, в которых встречаются сдвоенные буквы.	Функция, создающая массив чисел, меньше чем M.	Функция, которая удаляет из текста все предложения, в которых встречаются числа.

Лабораторная работа №7
Структуры и модули.

Цель работы:

Изучение принципов работы со строками в языке C. Использование параметров запуска для передачи данных в запускаемую программу.

Задание:

В данной лабораторной работе необходимо создать модуль, содержащий описание структуры данных, представляющее собой некоторую сущность: вектор, дробь, фигура и т.д. (по вариантам ниже), а также 8 операций (функций) над этой структурой. Основная программа должна запрашивать у пользователя какую операцию необходимо протестировать, далее запрашивать данные для операндов и показывать результат операции с помощью функции модуля «вывод в консоль». В качестве упрощения, задание не подразумевает использование динамической памяти, следовательно, во всех вариантах используются статические массивы (если это обусловлено задачей).

Важно: структуры создаются в модуле с помощью отдельной функции, возвращающей структуру по значению. Это необходимо, например, для того, чтобы рассчитать некоторые поля структуры на основе переданных данных.

Может показаться, что передача по значению локальной переменнойструктуры из функции неэффективно, т.к. происходит несколько раз объявление и копирование переменной. Однако в реальности не всё так плохо: в современных компиляторах присутствует NRVO-оптимизация (Named Return Value Optimization). Она заключается в том, что компилятор видя, что локальная переменная используется в качестве временного объекта для возвращаемого значения, выбросит создание этой переменной и будет работать уже непосредственно с той переменной, в результате к которой присваивается значение функции, что существенно повышает производительность. Без этой оптимизации была бы сначала создана локальная переменная-структура и проинициализирована значениями, затем при возврате значения функцией была бы создана копия данной структуры как временный объект, который затем опять был бы опять скопирован в созданную в основной программе переменную-структуру.

Важно: во всех функциях где написано «*результат возвращается как новое значение*» следует создавать и возвращать новую структуру, во всех других случаях следует модифицировать существующую, которая была передана в качестве параметра.

Варианты заданий:

Вариант 1. «Комплексное число» – Complex.

Разработать структуру данных Complex, представляющую комплексные числа в виде значений вещественной и мнимой части (a,b), а также логического значения real, показывающего что комплексное число имеет только вещественную часть (мнимая часть равна нулю). В модуле для работы с комплексными числами должны быть следующие функции:

1) Создание структуры комплексного числа из значений действительной и мнимой части. Как результат возвращается новое комплексное число.
2) Сложение двух комплексных чисел, как результат возвращается новое комплексное число.
3) Вычитание двух комплексных чисел, как результат возвращается новое комплексное число.

4) Умножение двух комплексных чисел, как результат возвращается новое комплексное число.

5) Деление двух комплексных чисел, как результат возвращается новое комплексное число.

6) Преобразование переданного комплексного числа в сопряжённое.

7) Преобразование переданного комплексного числа в обратное.

8) Вывод в консоль переданного комплексного числа в виде:

(*a + b * i) [real],* где real имеет значение либо «*комплексное*», либо

«*вещественное*».

Вариант 2. «**Дробь**» – **Fraction.**

Разработать структуру данных Fraction, представляющую простую дробь в виде пары целых положительных чисел (*m,n*) а также отдельно логического значения, указывающего на знак дроби. В модуле для работы с дробями должны быть следующие функции:

1) Создание структуры дроби из значений числителя, знаменателя, а также знака. Как результат возвращается новая дробь.

2) Сложение двух дробей, как результат возвращается новая дробь.

3) Вычитание двух дробей, как результат возвращается новая дробь.

4) Умножение двух дробей, как результат возвращается новая дробь.

5) Деление двух дробей, как результат возвращается новая дробь.

6) Приведение двух переданных дробей к общему знаменателю.

7) Сравнение двух дробей, как результат возвращается целое значение: -1 «меньше», 1 «больше», 0 «равны».

8) Вывод в консоль переданной дроби в виде:

[sign] a / b,

где sign имеет значение либо «+», либо «-».

Вариант 3. «**Двумерный вектор**» – **Vector2d.**

Разработать структуру данных **Vector2d**, представляющую двумерный вектор в виде двух компонент (x,y), а также логического значения norm, показывающего нормализован ли данный вектор. В модуле для работы с двумерными векторами должны быть следующие функции:

1) Создание структуры двумерного вектора из значений его компонент. Как результат возвращается новый вектор.
2) Сложение двух векторов, как результат возвращается новый вектор.
3) Вычитание двух векторов, как результат возвращается новый вектор.
4) Умножение переданного вектора на заданный скаляр.
5) Нормализация переданного вектора.
6) Получение значения скалярного произведения двух векторов.
7) Установка заданной длины для переданного вектора.
8) Вывод в консоль переданного вектора в виде:

(x,y) [norm], где norm имеет значение либо «*нормализован*», либо «*не нормализован*».

Вариант 4. «Трёхмерный вектор» – Vector3d.

Разработать структуру данных **Vector3d**, представляющую трёхмерный вектор в виде трех компонент (x,y,z), а также логического значения norm, показывающего нормализован ли данный вектор. В модуле для работы с трёхмерными векторами должны быть следующие функции:

1) Создание структуры трёхмерного вектора из значений его компонент. Как результат возвращается новый вектор.
2) Сложение двух векторов, как результат возвращается новый вектор.
3) Вычитание двух векторов, как результат возвращается новый вектор.
4) Получение значения угла между двумя переданными векторами.
5) Нормализация переданного вектора.
6) Векторное произведение двух векторов, как результат возвращается новый вектор.
7) Проекция вектора на вектор, как результат возвращается новый вектор.

8) Вывод в консоль переданного вектора в виде:

(x,y,z) [norm], где norm имеет значение либо «*нормализован*», либо «*не нормализован*».

Вариант 5. «Квадратная матрица 3x3» – Matrix3x3.

Разработать структуру данных **Matrix3x3**, представляющую квадратную матрицу 3 на 3 в виде массива из 9 её элементов, а также логического значения identity, показывающего является ли данная матрица единичной. В модуле для работы с матрицами должны быть следующие функции:

1) Создание структуры матрицы из массива значений её элементов. Как результат возвращается новая матрица.
2) Сложение двух матриц, как результат возвращается новая матрица.
3) Умножение двух матриц, как результат возвращается новая матрица.
4) Умножение переданной матрицы на скаляр.
5) Преобразование переданной матрицы в транспонированную.
6) Нахождение определителя переданной матрицы.
7) Преобразование переданной матрицы в обратную.
8) Вывод в консоль переданной матрицы в виде:

-1.4 3 2.5

3.2 1 -0.4

3.2 4.3 4

[identity]

где identity имеет значение либо «единичная», либо «не единичная»

Вариант 6. «Квадратная матрица 4x4» – Matrix4x4.

Разработать структуру данных **Matrix4x4**, представляющую квадратную матрицу 4 на 4 в виде массива из 16 её элементов, а также логического значения zero, показывающего является ли данная матрица нулевой. В модуле для работы с матрицами должны быть следующие функции:

1) Создание структуры матрицы из массива значений её элементов. Как результат возвращается новая матрица.

2) Сложение двух матриц, как результат возвращается новая матрица.

3) Умножение двух матриц, как результат возвращается новая матрица.

4) Умножение переданной матрицы на скаляр.

5) Нахождение тройки значений соответственно минимального, максимального и среднего значений элементов переданной матрицы.

6) Нахождение определителя переданной матрицы.

7) Преобразование переданной матрицы в обратную.

8) Вывод в консоль переданной матрицы в виде:

-1.4 3 2.5 3.2

3.2 1 -0.4 -3

3.2 4.3 4 1.2

-4 3.1 -3.4 2

[zero] где zero имеет значение либо «нулевая», либо «не

нулевая*»*

Вариант 7. «Прямоугольник» – Rectangle.

Разработать структуру данных Rectangle, представляющую фигуру выровненного по осям прямоугольника, заданный двумя координатами: соответственно верхней левой и нижней правой вершины, а также логическим значением square, показывающим является ли данный прямоугольник квадратом. В модуле для работы с прямоугольниками должны быть следующие функции:

1) Создание структуры прямоугольника по координатам его двух вершин. Как результат возвращается новый прямоугольник.

2) Перемещение переданного прямоугольника на заданное смещение по оси X и Y.

3) Изменение ширины и высоты для переданного прямоугольника.

4) Вычисление пары значений – площади и периметра – для переданного прямоугольника.

5) Нахождение прямоугольника минимальной площади, внутрь которого попадают все точки, переданные в качестве массива. Как результат возвращается новый прямоугольник.

6) Нахождение прямоугольника, как пересечение двух других переданных прямоугольников. Как результат возвращается новый прямоугольник.

7) Нахождение прямоугольника, вписанного в заданную окружность с координатами центра x,y и радиусом R. Как результат возвращается новый прямоугольник.

8) Вывод в консоль переданного прямоугольника в виде:

(0, 5) (10.5, 5) (0,0)

(10.5,0)

[square] где square имеет значение либо «квадрат», либо «прямоугольник».

Вариант 8. «Круг» – Circle.

Разработать структуру данных Circle, представляющую фигуру круг, заданный координатами его центра и радиусом. В модуле для работы с кругами должны быть следующие функции:

1) Создание структуры круга по координатам его центра и радиусу. Как результат возвращается новый круг.

2) Определение попадания заданной точки в переданный круг.

3) Определение факта пересечения двух переданных кругов.

4) Определение факта пересечения переданного круга с прямоугольной областью с параметрами (x,y,width,height).

5) Рассчитать площадь пересечения двух переданных кругов.

6) Рассчитать круг, описанный вокруг треугольника, заданный тремя координатами вершин. Как результат возвращается новый круг.

7) Рассчитать круг минимального радиуса, внутрь которого попадают все круги, переданные в качестве массива. Как результат возвращается новый круг.

8) Вывод в консоль переданного прямоугольника в виде: (x,y) [R]

Вариант 9. «Треугольник» - Triangle.

Разработать структуру данных Triangle, представляющую фигуру треугольник, заданный координатами его вершин, а также логическим значением equilateral, показывающим является ли данный треугольник равносторонним. В модуле для работы с треугольниками должны быть следующие функции:

1) Создание структуры треугольника по координатам его вершин. Как результат возвращается новый треугольник.
2) Перемещение переданного треугольника на заданное смещение относительно осей X и Y.
3) Расчёт координат центроида переданного треугольника.
4) Поворот переданного треугольника на заданный угол вокруг его центроида.
5) Изменение размера переданного треугольника на заданный коэффициент масштаба относительно его центроида.
6) Создание равностороннего треугольника по заданному размеру стороны, центроид которого совпадает с точкой (0,0). Как результат возвращается новый треугольник.
7) Создание прямоугольного треугольника по двум заданным длинам катетов, совпадающих с положительными полуосями X и Y. Как результат возвращается новый треугольник.
8) Вывод в консоль переданного треугольника в виде:

(3, 9.5)

(2.3, 3.4) (8.6, 2.5)

[equilateral]

где equilateral имеет значение либо «равносторонний», либо «не равносторонний».

Вариант 10. «Многочлен» – Polynom.

Разработать структуру данных **Polynom**, представляющую многочлен от одной переменной вида $ax^n+bx^{n-1}+\ldots dx + e$ степени n (максимальная степень 10), заданный статическим массивом коэффициентов (e, d … b, a) и без знаковым целым числом, выражающее степень n. В модуле для работы с многочленами должны быть следующие функции:

1) Создание структуры многочлена по переданному массиву коэффициентов. Как результат возвращается новый многочлен.
2) Сложение двух многочленов, как результат возвращается новый многочлен.
3) Вычитание двух многочленов, как результат возвращается новый многочлен.
4) Умножение двух многочленов, как результат возвращается новый многочлен.
5) Деление двух многочленов, как результат возвращается новый многочлен (остаток отбрасывается).
6) Умножение переданного многочлена на число.
7) Расчёт значения переданного многочлена для заданного значения переменной.
8) Вывод в консоль переданного многочлена в виде: a * x^n + b * x^n–1 + … + d * x + e

Вариант 11. «Множество целых чисел» – Set.

Разработать структуру данных Set, представляющую множество целочисленных элементов (с максимальным количеством элементов 50), представленной статическим массивом элементов, а также целочисленной переменной size, задающее текущий размер множества. В модуле для работы с множествами должны быть следующие функции:

1) Создание структуры множества по переданному массиву элементов. Как результат возвращается новое множество.
2) Принадлежность заданного элемента переданному множеству.
3) Добавление элемента к переданному множеству.
4) Удаление элемента из переданного множества.
5) Создать множество, являющееся пересечением двух переданных множеств. Как результат возвращается новое множество.
6) Создать множество, являющееся объединением двух переданных множеств. Как результат возвращается новое множество.
7) Создать множество, являющееся вычитанием двух переданных множеств. Как результат возвращается новое множество.

8) Вывод в консоль переданного множества в виде:

(a,b,c,d,e…) [size] где size –

размер множества.

Вариант 12. «Битовая матрица» – BitMatrix.

Разработать структуру данных **BitMatrix**, представляющую собой матрицу битовых значений представленной статическим массивом *целочисленных значений типа long long (8 байт), задающий строки данной матрицы, а также две целочисленные переменные, задающие текущий размер данной матрицы sizeX* и *sizeY (максимальный размер матрицы 64 на 64 бит)*. В модуле для работы с матрицами должны быть следующие функции:

1) Создание структуры битовой матрицы по переданному двумерному массиву элементов типа bool. Как результат возвращается новая матрица.
2) Получение значения бита с заданными индексами в переданной матрице как значение типа bool.
3) Логическое сложение двух равных по размеру матриц. Как результат возвращается новая матрица.
4) Логическое умножение двух равных по размеру матриц. Как результат возвращается новая матрица.
5) Инверсия переданной матрицы.
6) установка для переданной матрицы определённого логического значения для сегмента, заданного как (x,y,width,height).
7) Вычисление количества 0 и 1 в переданной матрице.
8) Вывод в консоль переданной матрицы в виде:

1, 0, 1, 1, 0, 1

1, 1, 1, 0, 0, 1

0, 0, 0, 1, 1, 1 [sizeX, sizeY] где sizeX и sizeY

– текущий размер матрицы.

Вариант 13. «Массив бит» – BitArray.

Разработать структуру данных **BitArray**, представляющую собой массив битовых значений представленный без знаковым long long значением (т.е. максимальной длины 64 бит), хранящим биты и целочисленной переменной size, хранящий текущий размер массива. В модуле для работы с массивом должны быть следующие функции:

1) Создание структуры битового массива по переданному массиву элементов типа bool. Как результат возвращается новый массив.
2) Установка значения бита на заданной позиции переданного массива.
3) Получение значения бита на заданной позиции переданного массива как логическое значение типа bool.
4) Выделение подмассива из переданного массива начиная с i-го бита и длиной n бит. Как результат возвращается новый массив.
5) Циклический сдвиг битов переданного массива вправо\влево на заданное число позиций (если значение сдвига положительное – то вправо, отрицательное – то влево).
6) Логическое сложение двух равных по размеру массивов. Как результат возвращается новый массив.
7) Логическое умножение двух равных по размеру массивов. Как результат возвращается новый массив.
8) Вывод в консоль переданного массива в виде:

1, 0, 1, 1, 0, 1 [size]

где size – текущий размер массива.

Вариант 14. «Комплексное число» – Complex.

Разработать структуру данных Complex, представляющую комплексные числа в виде значений вещественной и мнимой части (a,b), а также логического значения real, показывающего что комплексное число имеет только вещественную часть (мнимая часть равна нулю). В модуле для работы с комплексными числами должны быть следующие функции:

1) Создание структуры комплексного числа из значений действительной и мнимой части. Как результат возвращается новое комплексное число.
2) Сложение двух комплексных чисел, как результат возвращается новое комплексное число.

3) Вычитание двух комплексных чисел, как результат возвращается новое комплексное число.

4) Умножение двух комплексных чисел, как результат возвращается новое комплексное число.

5) Деление двух комплексных чисел, как результат возвращается новое комплексное число.

6) Преобразование переданного комплексного числа в сопряжённое.

7) Преобразование переданного комплексного числа в обратное.

8) Вывод в консоль переданного комплексного числа в виде:

(*a + b * i) [real],* где real имеет значение либо «*комплексное*», либо

«*вещественное*».

Вариант 15. «**Дробь**» – **Fraction.**

Разработать структуру данных Fraction, представляющую простую дробь в виде пары целых положительных чисел (*m*,*n*) а также отдельно логического значения, указывающего на знак дроби. В модуле для работы с дробями должны быть следующие функции:

1) Создание структуры дроби из значений числителя, знаменателя, а также знака. Как результат возвращается новая дробь.

2) Сложение двух дробей, как результат возвращается новая дробь.

3) Вычитание двух дробей, как результат возвращается новая дробь.

4) Умножение двух дробей, как результат возвращается новая дробь.

5) Деление двух дробей, как результат возвращается новая дробь.

6) Приведение двух переданных дробей к общему знаменателю.

7) Сравнение двух дробей, как результат возвращается целое значение: -1 «меньше», 1 «больше», 0 «равны».

8) Вывод в консоль переданной дроби в виде:

[sign] a / b,

где sign имеет значение либо «+», либо «-».

Вариант 16. «**Двумерный вектор**» – **Vector2d.**

Разработать структуру данных **Vector2d**, представляющую двумерный вектор в виде двух компонент (x,y), а также логического значения norm, показывающего нормализован ли данный вектор. В модуле для работы с двумерными векторами должны быть следующие функции:

1) Создание структуры двумерного вектора из значений его компонент. Как результат возвращается новый вектор.
2) Сложение двух векторов, как результат возвращается новый вектор.
3) Вычитание двух векторов, как результат возвращается новый вектор.
4) Умножение переданного вектора на заданный скаляр.
5) Нормализация переданного вектора.
6) Получение значения скалярного произведения двух векторов.
7) Установка заданной длины для переданного вектора.
8) Вывод в консоль переданного вектора в виде:

(x,y) [norm], где norm имеет значение либо «*нормализован*», либо «*не нормализован*».

Вариант 17. «**Трёхмерный вектор**» – **Vector3d.**

Разработать структуру данных **Vector3d**, представляющую трёхмерный вектор в виде трех компонент (x,y,z), а также логического значения norm, показывающего нормализован ли данный вектор. В модуле для работы с трёхмерными векторами должны быть следующие функции:

1) Создание структуры трёхмерного вектора из значений его компонент. Как результат возвращается новый вектор.
2) Сложение двух векторов, как результат возвращается новый вектор.
3) Вычитание двух векторов, как результат возвращается новый вектор.
4) Получение значения угла между двумя переданными векторами.
5) Нормализация переданного вектора.
6) Векторное произведение двух векторов, как результат возвращается новый вектор.

7) Проекция вектора на вектор, как результат возвращается новый вектор.

8) Вывод в консоль переданного вектора в виде:

(x,y,z) [norm], где norm имеет значение либо «*нормализован*», либо «*не нормализован*».

Вариант 18. «**Квадратная матрица 3x3**» – **Matrix3x3**.

Разработать структуру данных **Matrix3x3**, представляющую квадратную матрицу 3 на 3 в виде массива из 9 её элементов, а также логического значения identity, показывающего является ли данная матрица единичной. В модуле для работы с матрицами должны быть следующие функции:

1) Создание структуры матрицы из массива значений её элементов. Как результат возвращается новая матрица.

2) Сложение двух матриц, как результат возвращается новая матрица.

3) Умножение двух матриц, как результат возвращается новая матрица.

4) Умножение переданной матрицы на скаляр.

5) Преобразование переданной матрицы в транспонированную.

6) Нахождение определителя переданной матрицы.

7) Преобразование переданной матрицы в обратную.

8) Вывод в консоль переданной матрицы в виде:

-1.4 3 2.5

3.2 1 -0.4

3.2 4.3 4

[identity]

где identity имеет значение либо «единичная», либо «не единичная»

Вариант 19. «**Квадратная матрица 4x4**» – **Matrix4x4**.

Разработать структуру данных **Matrix4x4**, представляющую квадратную матрицу 4 на 4 в виде массива из 16 её элементов, а также логического значения zero, показывающего является ли данная матрица нулевой. В модуле для работы с матрицами должны быть следующие функции:

1) Создание структуры матрицы из массива значений её элементов. Как результат возвращается новая матрица.

2) Сложение двух матриц, как результат возвращается новая матрица.

3) Умножение двух матриц, как результат возвращается новая матрица.

4) Умножение переданной матрицы на скаляр.

5) Нахождение тройки значений соответственно минимального, максимального и среднего значений элементов переданной матрицы.

6) Нахождение определителя переданной матрицы.

7) Преобразование переданной матрицы в обратную.

8) Вывод в консоль переданной матрицы в виде:

-1.4 3 2.5 3.2

3.2 1 -0.4 -3

3.2 4.3 4 1.2

-4 3.1 -3.4 2

[zero]

где zero имеет значение либо «нулевая», либо «не нулевая»

Вариант 20. «Прямоугольник» – Rectangle.

Разработать структуру данных Rectangle, представляющую фигуру выровненного по осям прямоугольника, заданный двумя координатами: соответственно верхней левой и нижней правой вершины, а также логическим значением square, показывающим является ли данный прямоугольник квадратом. В модуле для работы с прямоугольниками должны быть следующие функции:

1) Создание структуры прямоугольника по координатам его двух вершин. Как результат возвращается новый прямоугольник.

2) Перемещение переданного прямоугольника на заданное смещение по оси X и Y.

3) Изменение ширины и высоты для переданного прямоугольника.

4) Вычисление пары значений – площади и периметра – для переданного прямоугольника.

5) Нахождение прямоугольника минимальной площади, внутрь которого попадают все точки, переданные в качестве массива. Как результат возвращается новый прямоугольник.

6) Нахождение прямоугольника, как пересечение двух других переданных прямоугольников. Как результат возвращается новый прямоугольник.

7) Нахождение прямоугольника, вписанного в заданную окружность с координатами центра x,y и радиусом R. Как результат возвращается новый прямоугольник.

8) Вывод в консоль переданного прямоугольника в виде:

(0, 5) (10.5, 5)

(0,0) (10.5,0) [square]

где square имеет значение либо «квадрат», либо «прямоугольник».

Вариант 21. «Круг» – Circle.

Разработать структуру данных Circle, представляющую фигуру круг, заданный координатами его центра и радиусом. В модуле для работы с кругами должны быть следующие функции:

1) Создание структуры круга по координатам его центра и радиусу. Как результат возвращается новый круг.

2) Определение попадания заданной точки в переданный круг.

3) Определение факта пересечения двух переданных кругов.

4) Определение факта пересечения переданного круга с прямоугольной областью с параметрами (x,y,width,height).

5) Рассчитать площадь пересечения двух переданных кругов.

6) Рассчитать круг, описанный вокруг треугольника, заданный тремя координатами вершин. Как результат возвращается новый круг.

7) Рассчитать круг минимального радиуса, внутрь которого попадают все круги, переданные в качестве массива. Как результат возвращается новый круг.

8) Вывод в консоль переданного прямоугольника в виде:

(x,y) [R]

Вариант 22. «Треугольник» - Triangle.

Разработать структуру данных Triangle, представляющую фигуру треугольник, заданный координатами его вершин, а также логическим значением equilateral, показывающим является ли данный треугольник равносторонним. В модуле для работы с треугольниками должны быть следующие функции:

1) Создание структуры треугольника по координатам его вершин. Как результат возвращается новый треугольник.
2) Перемещение переданного треугольника на заданное смещение относительно осей X и Y.
3) Расчёт координат центроида переданного треугольника.
4) Поворот переданного треугольника на заданный угол вокруг его центроида.
5) Изменение размера переданного треугольника на заданный коэффициент масштаба относительно его центроида.
6) Создание равностороннего треугольника по заданному размеру стороны, центроид которого совпадает с точкой (0,0). Как результат возвращается новый треугольник.
7) Создание прямоугольного треугольника по двум заданным длинам катетов, совпадающих с положительными полуосями X и Y. Как результат возвращается новый треугольник.
8) Вывод в консоль переданного треугольника в виде:

(3, 9.5)

(2.3, 3.4) (8.6, 2.5)

[equilateral]

где equilateral имеет значение либо «равносторонний», либо «не равносторонний».

Вариант 23. «Многочлен» – Polynom.

Разработать структуру данных **Polynom**, представляющую многочлен от одной переменной вида $ax^n+bx^{n-1}+\ldots dx + e$ степени n (максимальная степень

10), заданный статическим массивом коэффициентов (e, d … b, a) и без знаковым целым числом, выражающее степень n. В модуле для работы с многочленами должны быть следующие функции:

1) Создание структуры многочлена по переданному массиву коэффициентов. Как результат возвращается новый многочлен.
2) Сложение двух многочленов, как результат возвращается новый многочлен.
3) Вычитание двух многочленов, как результат возвращается новый многочлен.
4) Умножение двух многочленов, как результат возвращается новый многочлен.
5) Деление двух многочленов, как результат возвращается новый многочлен (остаток отбрасывается).
6) Умножение переданного многочлена на число.
7) Расчёт значения переданного многочлена для заданного значения переменной.
8) Вывод в консоль переданного многочлена в виде:

a * x^n + b * x^n–1 + … + d * x + e

Вариант 24. «Множество целых чисел» – Set.

Разработать структуру данных Set, представляющую множество целочисленных элементов (с максимальным количеством элементов 50), представленной статическим массивом элементов, а также целочисленной переменной size, задающее текущий размер множества. В модуле для работы с множествами должны быть следующие функции:

1) Создание структуры множества по переданному массиву элементов. Как результат возвращается новое множество.
2) Принадлежность заданного элемента переданному множеству.
3) Добавление элемента к переданному множеству.
4) Удаление элемента из переданного множества.
5) Создать множество, являющееся пересечением двух переданных множеств. Как результат возвращается новое множество.
6) Создать множество, являющееся объединением двух переданных множеств.

Как результат возвращается новое множество.

7) Создать множество, являющееся вычитанием двух переданных множеств. Как результат возвращается новое множество.

8) Вывод в консоль переданного множества в виде:

(a,b,c,d,e…) [size] где size –

размер множества.

Вариант 25. «Битовая матрица» – BitMatrix.

Разработать структуру данных **BitMatrix**, представляющую собой матрицу битовых значений представленной статическим массивом *целочисленных значений типа long long (8 байт), задающий строки данной матрицы, а также две целочисленные переменные, задающие текущий размер данной матрицы sizeX* и *sizeY (максимальный размер матрицы 64 на 64 бит)*. В модуле для работы с матрицами должны быть следующие функции:

1) Создание структуры битовой матрицы по переданному двумерному массиву элементов типа bool. Как результат возвращается новая матрица.

2) Получение значения бита с заданными индексами в переданной матрице как значение типа bool.

3) Логическое сложение двух равных по размеру матриц. Как результат возвращается новая матрица.

4) Логическое умножение двух равных по размеру матриц. Как результат возвращается новая матрица.

5) Инверсия переданной матрицы.

6) установка для переданной матрицы определённого логического значения для сегмента, заданного как (x,y,width,height).

7) Вычисление количества 0 и 1 в переданной матрице.

8) Вывод в консоль переданной матрицы в виде:

1, 0, 1, 1, 0, 1

1, 1, 1, 0, 0, 1

0, 0, 0, 1, 1, 1 [sizeX, sizeY] где sizeX и sizeY

– текущий размер матрицы.

Вариант 26. «Массив бит» – BitArray.

Разработать структуру данных **BitArray**, представляющую собой массив битовых значений представленный без знаковым long long значением (т.е. максимальной длины 64 бит), хранящим биты и целочисленной переменной size, хранящий текущий размер массива. В модуле для работы с массивом должны быть следующие функции:

1) Создание структуры битового массива по переданному массиву элементов типа bool. Как результат возвращается новый массив.
2) Установка значения бита на заданной позиции переданного массива.
3) Получение значения бита на заданной позиции переданного массива как логическое значение типа bool.
4) Выделение подмассива из переданного массива начиная с i-го бита и длиной n бит. Как результат возвращается новый массив.
5) Циклический сдвиг битов переданного массива вправо\влево на заданное число позиций (если значение сдвига положительное – то вправо, отрицательное – то влево).
6) Логическое сложение двух равных по размеру массивов. Как результат возвращается новый массив.
7) Логическое умножение двух равных по размеру массивов. Как результат возвращается новый массив.
8) Вывод в консоль переданного массива в виде:

1, 0, 1, 1, 0, 1 [size]

где size – текущий размер массива.

Вариант 27. «Комплексное число» – Complex.

Разработать структуру данных Complex, представляющую комплексные числа в виде значений вещественной и мнимой части (a,b), а также логического значения real, показывающего что комплексное число имеет только вещественную часть (мнимая часть равна нулю). В модуле для работы с комплексными числами должны быть следующие функции:

1) Создание структуры комплексного числа из значений действительной и мнимой части. Как результат возвращается новое комплексное число.

2) Сложение двух комплексных чисел, как результат возвращается новое комплексное число.

3) Вычитание двух комплексных чисел, как результат возвращается новое комплексное число.

4) Умножение двух комплексных чисел, как результат возвращается новое комплексное число.

5) Деление двух комплексных чисел, как результат возвращается новое комплексное число.

6) Преобразование переданного комплексного числа в сопряжённое.

7) Преобразование переданного комплексного числа в обратное.

8) Вывод в консоль переданного комплексного числа в виде:

(*a + b * i) [real],* где real имеет значение либо «*комплексное*», либо

«*вещественное*».

Вариант 28. «**Дробь**» – **Fraction.**

Разработать структуру данных Fraction, представляющую простую дробь в виде пары целых положительных чисел (*m,n*) а также отдельно логического значения, указывающего на знак дроби. В модуле для работы с дробями должны быть следующие функции:

1) Создание структуры дроби из значений числителя, знаменателя, а также знака. Как результат возвращается новая дробь.

2) Сложение двух дробей, как результат возвращается новая дробь.

3) Вычитание двух дробей, как результат возвращается новая дробь.

4) Умножение двух дробей, как результат возвращается новая дробь.

5) Деление двух дробей, как результат возвращается новая дробь.

6) Приведение двух переданных дробей к общему знаменателю.

7) Сравнение двух дробей, как результат возвращается целое значение: -1 «меньше», 1 «больше», 0 «равны».

8) Вывод в консоль переданной дроби в виде:

[sign] a / b,

где sign имеет значение либо «+», либо «-».

Вариант 29. «**Двумерный вектор**» – **Vector2d.**

Разработать структуру данных **Vector2d**, представляющую двумерный вектор в виде двух компонент (x,y), а также логического значения norm, показывающего нормализован ли данный вектор. В модуле для работы с двумерными векторами должны быть следующие функции:

1) Создание структуры двумерного вектора из значений его компонент. Как результат возвращается новый вектор.
2) Сложение двух векторов, как результат возвращается новый вектор.
3) Вычитание двух векторов, как результат возвращается новый вектор.
4) Умножение переданного вектора на заданный скаляр.
5) Нормализация переданного вектора.
6) Получение значения скалярного произведения двух векторов.
7) Установка заданной длины для переданного вектора.
8) Вывод в консоль переданного вектора в виде:

(x,y) [norm], где norm имеет значение либо «*нормализован*», либо «*не нормализован*».

Вариант 30. «**Трёхмерный вектор**» – **Vector3d.**

Разработать структуру данных **Vector3d**, представляющую трёхмерный вектор в виде трех компонент (x,y,z), а также логического значения norm, показывающего нормализован ли данный вектор. В модуле для работы с трёхмерными векторами должны быть следующие функции:

1) Создание структуры трёхмерного вектора из значений его компонент. Как результат возвращается новый вектор.
2) Сложение двух векторов, как результат возвращается новый вектор.
3) Вычитание двух векторов, как результат возвращается новый вектор.
4) Получение значения угла между двумя переданными векторами.

5) Нормализация переданного вектора.

6) Векторное произведение двух векторов, как результат возвращается новый вектор.

7) Проекция вектора на вектор, как результат возвращается новый вектор.

8) Вывод в консоль переданного вектора в виде:

(x,y,z) [norm], где norm имеет значение либо «*нормализован*», либо «*не нормализован*».

Вариант 31. «**Квадратная матрица 3x3**» – **Matrix3x3.**

Разработать структуру данных **Matrix3x3**, представляющую квадратную матрицу 3 на 3 в виде массива из 9 её элементов, а также логического значения identity, показывающего является ли данная матрица единичной. В модуле для работы с матрицами должны быть следующие функции:

1) Создание структуры матрицы из массива значений её элементов. Как результат возвращается новая матрица.

2) Сложение двух матриц, как результат возвращается новая матрица.

3) Умножение двух матриц, как результат возвращается новая матрица.

4) Умножение переданной матрицы на скаляр.

5) Преобразование переданной матрицы в транспонированную.

6) Нахождение определителя переданной матрицы.

7) Преобразование переданной матрицы в обратную.

8) Вывод в консоль переданной матрицы в виде:

-1.4 3 2.5

3.2 1 -0.4

3.2 4.3 4

[identity]

где identity имеет значение либо «единичная», либо «не единичная»

Вариант 32. «**Квадратная матрица 4x4**» – **Matrix4x4.**

Разработать структуру данных **Matrix4x4**, представляющую квадратную матрицу 4 на 4 в виде массива из 16 её элементов, а также логического значения zero, показывающего является ли данная матрица нулевой. В модуле для работы с матрицами должны быть следующие функции:

1) Создание структуры матрицы из массива значений её элементов. Как результат возвращается новая матрица.
2) Сложение двух матриц, как результат возвращается новая матрица.
3) Умножение двух матриц, как результат возвращается новая матрица.
4) Умножение переданной матрицы на скаляр.
5) Нахождение тройки значений соответственно минимального, максимального и среднего значений элементов переданной матрицы.
6) Нахождение определителя переданной матрицы.
7) Преобразование переданной матрицы в обратную.
8) Вывод в консоль переданной матрицы в виде:

-1.4 3 2.5 3.2

3.2 1 -0.4 -3

3.2 4.3 4 1.2

-4 3.1 -3.4 2

[zero]

где zero имеет значение либо «нулевая», либо «не нулевая»

Вариант 33. «Прямоугольник» – Rectangle.

Разработать структуру данных Rectangle, представляющую фигуру выровненного по осям прямоугольника, заданный двумя координатами: соответственно верхней левой и нижней правой вершины, а также логическим значением square, показывающим является ли данный прямоугольник квадратом. В модуле для работы с прямоугольниками должны быть следующие функции:

1) Создание структуры прямоугольника по координатам его двух вершин. Как результат возвращается новый прямоугольник.

2) Перемещение переданного прямоугольника на заданное смещение по оси X и Y.

3) Изменение ширины и высоты для переданного прямоугольника.

4) Вычисление пары значений – площади и периметра – для переданного прямоугольника.

5) Нахождение прямоугольника минимальной площади, внутрь которого попадают все точки, переданные в качестве массива. Как результат возвращается новый прямоугольник.

6) Нахождение прямоугольника, как пересечение двух других переданных прямоугольников. Как результат возвращается новый прямоугольник.

7) Нахождение прямоугольника, вписанного в заданную окружность с координатами центра x,y и радиусом R. Как результат возвращается новый прямоугольник.

8) Вывод в консоль переданного прямоугольника в виде:

(0, 5) (10.5, 5) (0,0)

(10.5,0)

[square] где square имеет значение либо «квадрат», либо «прямоугольник».

Практическое занятие по теме № 1 Знакомство с языком С. Ассемблер и ассемблерные вставки.

Вопросы:

1) Что такое языки программирования «высокого» и «низкого» уровня?
2) Что такое уровни абстракции?
3) Каково назначение языка С?
4) Какие есть команды для компиляции и выполнения программы в среде Dev C++?
5) Зачем нужна функция main?
6) Что такое ассемблер?
7) Что такое регистры процессора?
8) Что такое команда пересылки и как осуществить сложение и вычитание в ассемблере?
9) Что такое команда пересылки и как производится умножение и деление в ассемблере?

10) Что такое ассемблерная вставка? Для чего она нужна и как ещё добавить?
11) Что такое расширенная ассемблерная вставка? Формат записи?
12) Формат объявления констант и переменных в языке С?
13) Что такое область видимости в языке С? Какие области видимости бывают?
14) Какие есть целочисленные типы данных в языке С?
15) Какие есть вещественные типы данных в языке С?

Практическое занятие по теме № 2
Управляющие структуры языка С.

Вопросы:

1) Что такое автоматическое приведение типов? Каковы правила автоматического приведения типов?

2) Что такое явное преобразование типа и как оно осуществляется? 3) Формат функции printf? Что такое специальные символы? Приведите примеры.

4) Формат функции scanf? Что такое спецификаторы формата? Приведите примеры.
5) Что такое унарные, бинарные и тернарные операторы? Приведите примеры.
6) Что такое выражение? Что такое приоритет операций?
7) Какие существуют основные математические функции?
8) Формат функций rand и srand? Для чего они нужны?
9) Что такое пустой оператор? Что такое составной оператор?
10) Формат условного оператора?
11) Формат оператора выбора?
12) Формат цикла с постусловием?
13) Формат цикла с предусловием?
14) Формат цикла с параметром?
15) Для чего нужны операторы break и continue?

Практическое занятие по теме № 3
Массивы и функции языка С.

Вопросы:

1) Что такое массив? Как происходит обращение к элементам массива?
2) Какими понятиями характеризуется массив?
3) Формат объявления и инициализации массивов?

4) Приведите пример объявления и инициализации одномерного массива. 5) Приведите пример объявления и инициализации двумерного массива 6) Что такое функция?
7) Что такое чистая функция?
8) Формат объявления и вызова функции?
9) Что нужно сделать, когда вызов функции предшествует её объявлению? 10) Как объявить функцию, не принимающую аргументов или не возвращающую значение?
11) Для чего применяется ключевое слово inline?
12) Что такое константные параметры функции и для чего применяются?
13) Как передаётся одномерный массив в функцию через параметр?
14) Приведите пример передачи одномерного массива в функцию.
15) Приведите пример передачи двумерного массива в функцию.

Практическое занятие по теме № 4
Указатели.

Вопросы:

1) Что такое указатель?
2) Как объявляется указатель?
3) Какие операции определены для указателей?
4) Что такое нулевой указатель и для чего он применяется?
5) Какие существуют способы передачи параметров функции? В каких случаях какой способ необходимо выбирать?
6) Что такое константный указатель и для чего он применяется?
7) Как связаны массивы и указатели?
8) Как представляются одномерные массивы в памяти компьютера?
9) Как представляются двумерные массивы в памяти компьютера?
10) Что такое индексация массива с точки зрения указателей?
11) Как использовать указатели для обращения к элементам массива?
12) Как передавать массив в функцию через указатель?
13) Что такое пустой указатель и для чего он применяется?
14) Как объявить указатель на функцию? Для чего он может применяться? 15) Как и для чего можно использовать пустой указатель при объявлении указателей на функции?

Практическое занятие по теме № 5
Динамическая память.

Вопросы:

1) На какие области делится память, выделяемая для процесса?
2) Какие способы хранения объектов в памяти существуют?
3) Что такое динамическая память?
4) Какие преимущества даёт динамическое выделение памяти?
5) Какие преимущества существуют у статического выделения памяти?
6) Формат и назначение функции malloc?
7) Формат и назначение функции free?
8) Как создать одномерный динамический массив указанного размера?
9) Как функция free «узнаёт» о размере освобождаемого участка памяти?
10) Формат и назначение функции calloc?
11) Формат и назначение функции realloc?
12) Формат и назначение функции memcpy?
13) Формат и назначение функции memset?
14) Как можно создать двумерный динамический массив?
15) Что такое «утечка памяти»?

Практическое занятие по теме № 6
Строки и параметры запуска.

Вопросы:

1) Что такое строковый тип данных?
2) Как задаётся строковая константа, а как отдельный символ?
3) Как хранится строка в памяти компьютера?
4) Как используется указатель на символьный тип?
5) Как задаётся массив символьных строк?
6) Каков формат и назначение функций puts и gets и в чём их отличие от printf и scanf?
7) Каков формат и назначение функции fgets?
8) Каков формат и назначение функций strcat и strcpy?
9) Каков формат и назначение функций strcmp и strlen?
10) Каков формат и назначение функций strset и sprintf?
11) Каков формат и назначение функций atoi и atof?
12) Каков формат и назначение функций strchr и toupper\tolower?
13) Что такое параметры запуска и как их можно задавать?
14) Как в программе получить доступ к параметрам запуска?
15) Что такое компиляция программ и зачем она нужна?
16) Что такое этап препроцессинга и зачем он нужен?

17) Что такое этап компиляции и зачем он нужен?
18) Что такое этап ассемблирования и зачем он нужен?
19) Что такое этап компоновки и зачем он нужен?
20) Что такое объектный файл и для чего он нужен?
21) Что такое таблица символов и для чего она нужна?

Практическое занятие по теме № 7
Структуры и модули.

Вопросы:

1) Что такое структура и для чего она нужна?
2) Что такое поля структуры?
3) Как происходит объявление структуры?
4) Как происходит копирование структуры?
5) Как использовать указатель на структуру и как обращаться к полям указателя на структуру?
6) Какими способами можно передавать структуру в качестве аргумента функции и в чём отличие между ними?
7) Что такое перечисление и для чего оно нужно?
8) Как можно объявить перечисление?
9) Что такое побитовая операция?
10) Какие побитовые операции существуют? Приведите примеры. 11) Что такое битовые флаги и в каких случаях они используются? Приведите примеры.
12) Что такое битовые маски и в каких случаях они используются?

Приведите примеры.

13) Что такое битовые поля, в чём их преимущество и в каких случаях они нужны?

Библиографический список

1. Керниган Б., Ритчи Д. Язык программирования Си. — 2-е изд. — М.: Вильямс, 2007. — С. 304. — ISBN 0-13-110362-8. 2. Гукин Д. Язык программирования Си. — М.: Диалектика, 2006. — С. 352. — ISBN 0-7645-7068-4.
3. Подбельский В. В., Фомин С. С. Курс программирования на языке Си: учебник. — М.: ДМК Пресс, 2012. — 318 с. — ISBN 978-5-94074-449-8.
4. Прата С. Язык программирования С: Лекции и упражнения. — М.: Вильямс, 2006. — С. 960. — ISBN 5-8459-0986-4.

5. Прата С. Язык программирования С (С11). Лекции и упражнения, 6-е издание. — М.: Вильямс, 2015. — 928 с. — ISBN 978-5-8459-1950-2. 6. Столяров А. В. Язык Си и начальное обучение программированию. — Издательский отдел факультета ВМК МГУ, 2010. — № 7. — С. 78—90.
7. Шилдт Г. С: полное руководство, классическое издание. — М.: Вильямс, 2010. — С. 704. — ISBN 978-5-8459-1709-6.
8. Языки программирования Ада, Си, Паскаль / А. Фьюэр, Н. Джехани. — М.: Радио и Саязь, 1989. — 368 с. — 50 000 экз. — ISBN 5-256-00309-7.

Составил: к.т.н., доцент кафедры
«Вычислительная и прикладная математика» Антипов О.В.

Заведующий кафедрой вычислительной и
прикладной математики, д-р техн. наук, профессор Овечкин Г.В.