Министерство образования и науки Российской Федерации
Государственное образовательное учреждение
высшего профессионального образования
«Кузбасский государственный технический университет»

Кафедра информационных и автоматизированных
производственных систем

ПРОГРАММИРОВАНИЕ И ОСНОВЫ АЛГОРИТМИЗАЦИИ

Методические указания к лабораторным работам для студентов специальности 220301 «Автоматизация технологических процессов и производств (в машиностроении)»

Составитель В. А. Полетаев

Утверждены на заседании кафедры
Протокол № 5 от 02.02.2011

Рекомендованы к печати
учебно-методической комиссией
специальности 220301
Протокол № 322 от 10.03.2011

Электронная копия находится
в библиотеке ГУ КузГТУ

Кемерово 2011

ВВЕДЕНИЕ

Язык С – это язык программирования общего назначения, хорошо известный своей эффективностью, экономичностью, и переносимостью. Преимущества С обеспечивают хорошее качество разработки почти любого вида программного продукта. Использование С в качестве инструментального языка позволяет получать быстрые и компактные программы.

Методические указания для проведения лабораторных работ по дисциплине "Программирование и основы алгоритмизации" включают краткие теоретические материалы, примеры решения задач, а также задачи для программирования, ориентированные на изучение программирования линейных, ветвящихся, циклических алгоритмов с использованием основных синтаксических конструкций языка С и С++.

Лабораторная работа 1.
Программирование линейных алгоритмов

1. ЦЕЛЬ РАБОТЫ

Целью работы является приобретение навыков программирования линейных алгоритмов.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Программы с линейной структурой составляются из операторов присваивания, ввода, вывода, обращения к процедурам.

Оператор присваивания можно назвать основным в любом языке программирования.

Оператор присваивания: <переменная>=<выражение>

Оператор выполняется следующим образом. Вычисляется значение <выражения>, после чего <переменная> получает

вычисленное значение. При этом тип выражения должен быть совместим с типом переменной.

Выражение может включать в себя константы, переменные, знаки операций, функции, скобки. В результате вычисления вы-

ражения получается значение определенного типа. Тип выражения определяется типом полученного значения.

Арифметическое выражение – выражение числового типа (целого или дробного). Идентификатор целого типа: **int**, дробного типа **float** или **double**.

Арифметические операции. К арифметическим операциям языка С относятся: – вычитание и унарный минус; + сложение; * умножение; / деление; % деление по модулю; ++ увеличение на единицу; – – уменьшение на единицу.

Стандартные математические функции языка С описаны в библиотеке **math.h.** Основные функции представлены в табл. 1.

Таблица 1

Основные математические функции языка С

Обращение	Функция	Обращение	Функция
fabs(x)	Модуль аргумента	**log(x)**	Логарифм натуральный
tan(x)	Тангенс аргумента (x в рад.)	**log10(x)**	Логарифм десятичный
cos(x)	Косинус аргумента	**pow(x,y)**	Возведение в степень x^y
sin(x)	Синус аргумента	**exp(x)**	Экспонента e^x
sqrt(x)	Корень квадратный	**cosh(x)**	Косинус гиперболический

Пример 1. Записать математические выражения в виде арифметических выражения языка С (табл. 1).

Таблица 1

Математическое выражение		Выражение на языке С	
1.	$x^2 - 7x + 6$	1.	**x*x – 7*x+6**
2.	$\frac{\lvert x\rvert - \lvert y\rvert}{1 + \lvert x + y\rvert}$	2.	**(fabs(x) – fabs(y))/(1+fabs(x*y))**

Ввод данных с клавиатуры и их вывод на дисплей производится путем обращения к стандартным функциям ввода/вывода, описанным в библиотеке: **stdio.h.** Функция форматного ввода **scanf(),** вывода **printf().**

Пример 2.

scanf("%d",&x)

где %d – формат вводимого числа (%d – целое десятичное число типа int; %c – символ типа char; %lf – число типа double и т.д.); & – операция взятия адреса; x – имя вводимой переменной.

printf("Число равно %d",x)

где %d – формат выводимого числа; x – имя выводимой переменной; "Число равно" – произвольный текст.

Структура программы на языке С:

```
# include<имя файла(библиотеки)>
<описание макроопределений>
<описание функций>
<объявление глобальных переменных>
main()
{ <объявление локальных переменных>
<тело программы>
return; }
```

Пример 3. Выполнить вычисление по формуле.

```
# include<stdio.h>
# include<math.h>
main()
{           double x,y,z;
            printf("Введите x и y");
            scanf("%lf%lf",&x,&y);
            z=fabs(x*x+y*y);
            printf("z равно %lf",z);
            return;
}
```

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить у преподавателя номер варианта задания.
2. Составить алгоритм решения заданий № 1, 2.
3. Написать программы на языке С, реализующие алгоритмы п. 2.
4. Оформить отчет о работе.

4. ВАРИАНТЫ ЗАДАНИЙ

Задание 1.

Вычислить значение функции по заданному значению переменных (табл. 2)

Таблица 2

№	Вид формулы	№	Вид формулы
1	$3^{-x} - \cos x + \sin(2xy)$	2	$x - 10^{\sin x} + \cos(x-y)$
3	$\frac{\sin x + \cos y}{\cos x - \sin y} \operatorname{tg} xy$	4	$\left(\frac{x+1}{x-1}\right)^x + 18xy^2$
5	$\left(\frac{\ln\lvert\cos x\rvert}{\ln(1+x^2)}\right)$	6	$\lvert x^2 - x^3\rvert - \frac{7x}{x^3 - 15x}$
7	$\left(1+\frac{1}{x^2}\right)^x - 12x^2y$	8	$x - 10\sin x + \lvert x^4 - x^5\rvert$
9	$\frac{\cos x}{\pi - 2x} + 16x\cos(xy) - 2$	10	$2\operatorname{ctg}(3x) - \frac{1}{12x^2 + 7x - 5}$
11	$3^x - 4x + (y - \sqrt{\lvert x\rvert})$	12	$\sin\sqrt{x+1} - \sin\sqrt{x-1}$
13	$\frac{b + \sqrt{b^2 + 4ac}}{2a} - a^3c + b^{-2}$	14	$\frac{a}{c}\frac{b}{d} - \frac{ab - c}{cd}$
15	$\frac{x+y}{x+1} - \frac{xy-12}{34+x}$	16	$x - \frac{x^3}{3} + \frac{x^5}{5}$

Задание 2.

1. Вычислить периметр и площадь прямоугольного треугольника по заданным длинам двух катетов a и b .

2. Заданы координаты трех вершин треугольника (x_1, y_1), (x_2, y_2), (x_3, y_3). Найти его периметр и площадь.

3. Вычислить длину окружности и площадь круга одного и того же заданного радиуса R .

4. Вычислить расстояние между двумя точками с данными координатами (x_1, y_1) и (x_2, y_2).

5. Даны два действительных числа x и y. Вычислить их сумму, разность, произведение и частное.

6. Дана длина ребра куба. Найти площадь грани, площадь полной поверхности и объем этого куба.

7. Дана сторона равностороннего треугольника. Найти площадь этого треугольника, его высоты, радиусы вписанной и описанной окружностей.

8. Известна длина окружности. Найти площадь круга, ограниченного этой окружностью.

9. Найти площадь кольца, внутренний радиус которого равен r, а внешний – заданному числу R ($R > r$).

10. Треугольник задан величинами своих углов и радиусом описанной окружности. Найти стороны треугольника.

11. Найти площадь равнобедренной трапеции с основаниями a и b и углом α при большем основании a.

12. Вычислить корни квадратного уравнения $ax^2 + bx + c = 0$, заданного коэффициентами a, b и c (предполагается, что $a \neq 0$ и что дискриминант уравнения неотрицателен).

13. Дано действительное число x. Не пользуясь никакими другими арифметическими операциями, кроме умножения, сложения и вычитания, вычислить за минимальное число операций $2x^4 - 3x^3 + 4x^2 - 5x + 6$.

14. Найти площадь треугольника, две стороны которого равны a и b , а угол между этими сторонами q.

15. Дано a. Не используя никаких функций и никаких операций, кроме умножения, получить a^8 за три операции; a^{10} и a^{16} за четыре операции.

16. Найти сумму членов арифметической прогрессии, если известны ее первый член, знаменатель и число членов прогрессии.

5. СОДЕРЖАНИЕ ОТЧЕТА

1. Ф.И.О. студента; № варианта;
2. Номер и название работы;
3. Цель работы;
4. Текст задания (постановка задачи);
5. Блок-схема алгоритма решения поставленной задачи;

6. Текст программы;
7. Результаты выполнения программы;
8. Выводы.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Использование оператора присваивания.
2. Приоритеты операций в языке С.
3. Использование стандартных математических функций.
4. Использование функций **scanf()** и **printf()**.
5. Структура программы на языке С.

Лабораторная работа 2.
Программирование ветвящихся алгоритмов

1. ЦЕЛЬ РАБОТЫ

Приобретение навыков программирования ветвящихся алгоритмов.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Для программирования ветвящихся алгоритмов применяются условный оператор и оператор выбора.

Условный оператор имеет следующий формат:

if(<логическое выражение>) <оператор 1>;
else <оператор 2>;

Операторы 1 и 2 могут быть простыми или составными. Если логическое выражение, выступающее в качестве условия ветвления, принимает значение 0 (ложно), то выполняется оператор 2, если 1 (истина) – оператор 1.

Неполная форма условного оператора:

if(<логическое выражение>) <оператор>;

Пример 1. Из трех данных вещественных чисел x, y, z выбрать наибольшее.

```
# include<stdio.h>
void main()
{           double x,y,z,max;
            printf("Введите x y z");
            scanf("%lf%lf%lf",&x,&y,&z);
            if(x>=y) if(x>=z) max=x;
            else max=z;
            else if(y>=z) max=y;
            else max=z;
            printf("Максимальное значение=%lf",max);
}
```

Оператор выбора позволяет программировать ветвления по многим направлениям. Этот оператор организует переход на одну из нескольких ветвей в зависимости от значения заданного выражения (селектора выбора).

Формат оператора выбора:

```
  switch(<выражение>)
{     case constant1: <последовательность операторов>; break;
      ….
      case constantN: <последовательность операторов>; break;
      default: <последовательность операторов>;
}
```

Условная операция имеет три операнда:

<условие>?<оператор 1>:<оператор 2>;

Вычисляется условие. Если условие истинно, то выполняется оператор 1 и его результат есть результат операции.

Пример 2. $c = (a + b) == 0? 3:5$; Если сумма a и b равна нулю, то c будет равно 3, иначе $c = 5$.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить у преподавателя номер варианта задания.
2. Составить алгоритм решения заданий.
3. Написать программы.

4. Оформить отчет о работе.

4. ВАРИАНТЫ ЗАДАНИЙ

1. Даны две точки $A(x_1, y_1)$ и $B(x_2, y_2)$. Составить алгоритм, определяющий, которая из точек находится ближе к началу координат.

2. Имеется пронумерованный список деталей: 1) шуруп, 2) гайка, 3) винт, 4) гвоздь, 5) болт. Составить программу, которая по номеру детали выводит на экран ее название.

3. Даны целые числа m, n . Если числа не равны, то заменить каждое из них одним и тем же числом, равным большему из исходных, а если равны, то заменить числа нулями.

4. Определить, равен ли квадрат заданного трехзначного числа кубу суммы цифр этого числа.

5. Написать программу, которая по номеру месяца выдает название следующего за ним месяца (при m = 1 получаем февраль, 4 – май).

6. Подсчитать количество отрицательных чисел среди чисел a, b, c.

7. Подсчитать количество целых чисел среди чисел a, b, c.

8. Определить, делителем каких чисел a, b, c является число k.

9. Перераспределить значения переменных x и y так, чтобы в x оказалось большее из этих значений, а в y – меньшее.

10. Определить правильность даты, введенной с клавиатуры (число – от 1 до 31, месяц – от 1 до 12). Если введены некорректные данные, то сообщить об этом.

11. Написать программу, распознающую по длинам сторон среди всех треугольников прямоугольные. Если их нет, то вычислить величину угла c.

12. Найти $\max\{\min(a, b), \min(c, d)\}$.

13. Даны четыре точки $A_1(x_1, y_1)$, A_2 (x_2, y_2), A_3 (x_3, y_3), A_4 (x_4, y_4). Определить, будут ли они вершинами параллелограмма.

14. Даны три точки $A(x_1, y_1)$, $B(x_2, y_2)$, C (x_3, y_3). Определить, будут ли они расположены на одной прямой. Если нет, то вычислить угол ABC .

15. Даны действительные числа a, b, c. Удвоить эти числа, если $a < b < c$, и заменить их абсолютными значениями, если это не так.

16. Для целого числа k от 1 до 99 напечатать фразу "Мне k лет", учитывая при этом, что при некоторых значениях k слово "лет" надо заменить на слово "год" или "года". Например, 11 лет, 22 года, 51 год.

5. СОДЕРЖАНИЕ ОТЧЕТА

1. Ф.И.О. студента; № варианта;
2. Номер и название работы;
3. Цель работы;
4. Текст задания (постановка задачи);
5. Блок-схема алгоритма решения поставленной задачи;
6. Текст программы;
7. Результаты выполнения программы;
8. Выводы.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Условный оператор. Полная и краткая форма.
2. Программирование логических выражений.
3. Использование оператора выбора.
4. Использование операторов **return** и **break**.
5. Применение условной операции.

Лабораторная работа 3.
Программирование циклических алгоритмов

1. ЦЕЛЬ РАБОТЫ

Приобретение навыков программирования циклических алгоритмов.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Цикл – многократное повторение последовательности действий по некоторому условию. Известны три типа циклических

алгоритмических структур: цикл с параметром, цикл с предусловием и цикл с постусловием.

Цикл с параметром имеет следующую форму записи:

for(<выражение 1>;<условие>;<выражение 2>) <тело цикла>;

Выражение 1 вычисляется один раз до начала выполнения цикла. Далее проверяется условие, если оно истинно, то выполняется тело цикла, а затем вычисляется выражение 2.

Пример 1. Вычислить сумму чисел от 1 до заданного числа *n*.

```
# include<stdio.h>
void main()
{           nt i,n,s;
            printf("Введите n");
            scanf("%d",&n);
            s=0;
            for(i=1;i<=n;i=i+1) s=s+i; или for(i=1;i<=n;s+=i++);
            printf("Сумма равна %d",s);
}
```

В качестве тела цикла может быть использован другой цикл (вложенный).

Пример 2. Вывести на экран таблицу умножения.

```
# include<stdio.h>
void main()
      {           int i,j;
                  for(i=1;i<=9;i++)
                  {     printf("\n");
      for(j=1;j<=9;j++) printf(" %d",i*j);
      }
}
```

Цикл с предусловием имеет следующий формат:

while(<условие>) <тело цикла>;

Вычисляется условие. Если оно истинно, то выполняется тело цикла и осуществляется переход в начало цикла для очередной проверки условия.

Пример 3. Вычислить сумму чисел от 1 до заданного числа *n*.

```
# include<stdio.h>
void main()
{       int a,s;
        printf("Введите n");
        scanf("%d",&n);
        s=0;
        a=1;
        while(a<=n)
        {       s=s+a;
a=a+1;
}
        printf("Сумма равна %d",s);
}
```

Цикл с постусловием имеет следующий формат:

do <тело цикла> **while**(<условие>);

Выполняется тело цикла и вычисляется условие. Если оно истинно, то осуществляется переход на начало цикла, а если нет, то выполняются последующие за циклом операторы.

Пример 4. Вычислить сумму чисел от 1 до заданного числа *n*.

```
# include<stdio.h>
void main()
{       int a,s;
        printf("Введите n");
        scanf("%d",&n);
        s=0;
        a=1;
do{
s=s+a;
a=a+1;
} while(a<=n);
        printf("Сумма равна %d",s);
```

}

Внутри цикла может использоваться оператор **continue,** который вызывает пропуск всех операторов до конца цикла с последующим продолжением цикла.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить у преподавателя номер варианта задания.
2. Составить алгоритм решения заданий.
3. Написать программы.
4. Оформить отчет о работе.

4. ВАРИАНТЫ ЗАДАНИЙ

1. Даны действительное число a, натуральное число n. Вычислить:

$$P = a\,(a - n)\,(a - 2n) \times \ldots \times (a - n^2\,).$$

2. Дан числовой ряд и малая величина ε . Найти сумму ряда с точностью ε, общий член которого задан формулой:

$$a_n = \frac{1}{2^n} + \frac{1}{3^n}.$$

3. Составить программу вычисления значений функции $F(x)$ на отрезке $[a; b]$ с шагом h . Результат представить в виде таблицы, первый столбец которой – значения аргумента, второй – соответствующие значения функции $F(x) = \sin 2x - \cos 2x$.

4. Дано натуральное число n. Найти сумму первой и последней цифры этого числа.

5. Среди всех n-значных чисел указать те, сумма цифр которых равна данному числу k.

6. Дано натуральное число n. Переставить местами первую и последнюю цифры этого числа.

7. Для записи римскими цифрами используются символы *I, V, X, L, C, D, M,* обозначающие соответственно числа 1, 5, 10, 50, 100, 500, 1000. Составить программу, которая запись любого данного числа n ($n \le 3999$) арабскими цифрами переводила бы в запись римскими цифрами.

8. Числа Фибоначчи (f_n) определяются формулами $f_0=f_1=1$, $f_n = f_{n-1} + f_{n-2}$ при $n = 2, 3, \ldots$. Определить f_{40} .

9. Дано натуральное число n. Переставить его цифры так, чтобы образовалось максимальное число, записанное теми же цифрами.

10. Составить программу, которая печатает таблицу умножения.

11. Покупатель должен заплатить в кассу S р. У него имеются 1, 2, 5, 10, 50, 100, 500 р. Сколько купюр разного достоинства отдаст покупатель, если он начинает платить с самых крупных?

12. Одноклеточная амеба каждые три 3 часа делится на 2 клетки. Определить, сколько амеб будет через 3, 6, 9, 12, …, 24 часа.

13. Составить программу, которая запрашивает пароль (четырехзначное число) до тех пор, пока количество неверных ответов не превысит три.

14. Найти все двузначные числа, сумма квадратов цифр которых кратна M .

15. Составить программу перевода данного натурального числа из десятичной системы счисления в двоичную.

16. Найти сумму всех n-значных чисел ($1 \le n \le 4$).

5. СОДЕРЖАНИЕ ОТЧЕТА

1. Ф.И.О. студента; № варианта;
2. Номер и название работы;
3. Цель работы;
4. Текст задания (постановка задачи);
5. Блок-схема алгоритма решения поставленной задачи;
6. Текст программы;
7. Результаты выполнения программы;
8. Выводы.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Использование цикла с параметром.
2. Программирование вложенных циклов.
3. Использование цикла с предусловием.

4. Использование цикла с постусловием.
5. Использование оператора **continue.**

Лабораторная работа 4.
Работа с массивами

1. ЦЕЛЬ РАБОТЫ

Приобретение навыков работы с массивами.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Массив – упорядоченный набор однотипных значений – компонент массива. Тип компонент называется базовым типом массив. Массив имеет имя, посредством имя, которого можно ссылаться на него как единое целое, так и на любую из его компонент.

Переменная с индексом – идентификатор компоненты массива. Формат записи:

<имя массива>[<индекс>],

где индекс может быть выражением целого типа.

Описание массива определяет базовый тип, имя массива и размер. Описание массива помещается в разделе объявления переменных и имеет вид:

<базовый тип> <имя>[<размер>];

Примеры объявления одномерного массива имеют вид:

int a[8]; – объявлен массив с именем a, состоящий из восьми элементов типа int;

int x,a[8],c,v; – массив объявляется наряду с объявлением переменных x, c, v;

int a[4]={3,8,2,6}; – непосредственное заполнение массива данными в момент объявления.

Примеры объявления многомерного массива имеют вид:

int a[3][4]; – объявлен двумерный массив с именем a размером 3 × 4, состоящий из элементов типа int;

int x,a[8][5],c,v; – массив объявляется наряду с объявлением переменных x, c, v;

int a[4][2]={{3,8},{2,6},{1,5},{8,5}}; – непосредственное заполнение массива данными в момент объявления;

int a[3][4][2]; – объявлен трехмерный массив с именем a размером 3× 4× 2, состоящий из элементов типа int.

Индексация элементов массива имеет значение 0 для первого элемента массива.

Ввод и вывод массива производится поэлементно. Обычно для этого используется цикл с параметром, где в качестве параметра применяется индексная переменная.

Пример 1. Ввести десять чисел, отсортировать их и вывести на дисплей.

```
# include<stdio.h>
void main()
{       int i,j,a[10],p;
        for(i=0;i<10;i++) scanf("%d",&a[i]);
        for(i=0;i<9;i++)
        {       for(j=0;j<9;j++)
        {       if(a[j]<=a[j+1])
        {       p=a[j];
                        a[j]=a[j+1];
                        a[j+1]=p;
        }
        }
        }
        printf("%d",a[i]);
}
```

Пример 2. Заполнить двумерный массив 10×10 значением 5.

```
# include<stdio.h>
void main()
{       int i,j,a[10][10];
        for(i=0;i<10);i++)
        for(j=0;j<10);j++) a[i][j]=5;
}
```

Динамическое определение массива происходит в ходе выполнения программы. Размер массива определяется значением некоторой переменной.

Пример 3. Работа с одномерным динамическим массивом.

```
# include<stdio.h>
# include<stdlib.h> – подключение библиотеки для динамического выделения памяти
void main()
{       int *p; – объявление указателя на динамический массив
        int t,i;
        printf("\nВведите размер массива");
        scanf("%d",&d); – определение размера будущего массива
        p=malloc(t*sizeof(int)); – размещение массива
        for(i=0;i<t;i++)
        {       a[i]=t*t; – работа с массивом
                printf("\n %d",a[i]); – вывод массива на дисплей
        }
        free(p); – освобождение памяти, занимаемой массивом
}
```

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить у преподавателя номер варианта задания.
2. Составить алгоритм решения заданий.
3. Написать программы.
4. Оформить отчет о работе.

4. ВАРИАНТЫ ЗАДАНИЙ

1. Составить программу, которая заполняет квадратную матрицу порядка n натуральными числами 1, 2, 3, …, n^2 , записывая их "по спирали".

2. Задана квадратная матрица. Переставить строку с максимальным элементом на главной диагонали со строкой с заданным номером m.

3. Вычислить сумму и число положительных элементов матрицы $A[N, N]$, находящихся над главной диагональю.

4. Дана целая квадратная матрица n-го порядка. Определить, является ли она магическим квадратом, т.е. такой, в которой суммы элементов во всех строках и столбцах одинаковы.

5. Дан целочисленный массив $A[n]$, среди элементов есть одинаковые. Создать массив из различных элементов $A[n]$.

6. Дан массив натуральных чисел. Найти сумму элементов, кратных данному K.

7. В целочисленной последовательности есть нулевые элементы. Создать массив из номеров этих элементов.

8. Дана последовательность целых чисел a_1, a_2, ..., a_n. Выяснить, будет ли она возрастающей.

9. Дана последовательность чисел a_1, a_2, ..., a_n. Заменить все ее члены, большие данного Z, этим числом. Подсчитать количество замен.

10. Дана последовательность чисел, среди которых имеется один ноль. Вывести на печать все числа, включительно до нуля.

11. Дана матрица A размера $n \times m$. Определить количество элементов массива больших заданного элемента массива.

12. Получить матрицу, в которой крайние элементы равны 1, а остальные 0.

13. Определить максимальный и минимальный элемент в матрице $m \times n$.

14. Дана действительная квадратная матрица порядка n. Преобразовать матрицу по правилу: строку с номером n сделать столбцом с номером n.

15. Упорядочить по возрастанию элементы каждой строки матрицы размером $n \times m$.

16. Дана действительная матрица порядка $2n$. Получить новую матрицу, переставляя ее блоки размера $n \times n$ крест - накрест.

5. СОДЕРЖАНИЕ ОТЧЕТА

1. Ф.И.О. студента; № варианта;
2. Номер и название работы;
3. Цель работы;
4. Текст задания (постановка задачи);
5. Блок-схема алгоритма решения поставленной задачи;
6. Текст программы;

7. Результаты выполнения программы;
8. Выводы.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Объявление одномерных и многомерных массивов.
2. Обращение к элементам массива.
3. Динамически размещаемые одномерные и многомерные массивы.
4. Освобождение памяти, занимаемой одномерными и многомерными динамическими массивами.
5. Понятия адреса и указателя.
6. Использование адресных операций.

Лабораторная работа 5.
Работа с функциями

1. ЦЕЛЬ РАБОТЫ

Приобретение навыков работы с функциями.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Функции – это блоки языка программирования, самостоятельные единицы программы, спроектированные для решения конкретных задач, обычно повторяющиеся несколько раз.

Объявление функций. Основная форма описания функции имеет вид:

тип <имя функции>(список параметров){<тело функции>}

Тип определяет тип значения, которое возвращает функция с помощью оператора **return**. Список параметров состоит из перечня типов и имен передаваемых параметров, разделенных запятыми. Функция может не иметь параметров, но круглые скобки необходимы в любом случае.

Пример 1. Выполнить вычисление по формуле, используя функцию: $z = x^2 + y^2$.

Как видно из примера 1 (табл. 3), описание функции может быть произведено до и после функции main, а объявление функции, т.е. задание ее прототипа необходимо производить до функции main. В примере 1 прототипом функции является строка **double fn(double x,double y).**

Таблица 3

Вариант 1	Вариант 2
# include<stdio.h> # include<math.h> double fn(double x,double y) {return fabs(x*x+y*y);} void main() {double x,y,z; printf("Введите x и y"); scanf("%lf%lf",&x,&y); z=fn(x,y); printf("z равно %lf",z); }	**# include<stdio.h> # include<math.h> double fn(double x,double y); void main() {double x,y,z; printf("Введите x и y"); scanf("%lf%lf",&x,&y); z=fn(x,y); printf("z равно %lf",z); } double fn(double x,double y) {return fabs(x*x+y*y);}**

В качестве аргумента (передаваемого значения) функции можно использовать массив. Одним из способов объявления такой функции является строка:

тип <имя функции>(<тип элементов массива> *<имя массива>);

Пример 2. Объявить функцию, в качестве аргумента которой используется массив с именем **ar,** состоящий из 10 элементов типа **int.** 1 способ: **void sort(int *ar);** 2 способ: **void sort(int ar[]);** 3 способ: **void sort(int ar[10]);**

Пример 3. Оформить сортировку массива в виде функции.

```
# include<stdio.h>
void sort(int *arr,int n);
void main()
{     int mass[10]={1,3,-5,7,9,0,22,4,6,8};
      int size=10,i;
      for(i=0;i<10;i++) printf(" %d",mass[i]);
      sort(mass,size);
```

```
        printf("\n");
        for(i=0;i<10;i++) printf(" %d",mass[i]);
}
void sort(int *arr,int n)
{       int i,j,tmp;
        for(i=0;i<n-1;i++)
        for(j=0;j<n-i-1;j++)
        if(arr[j+1]<arr[j])
        {       tmp=arr[j];
                        arr[j]=arr[j+1];
                        arr[j+1]=tmp;
        }
}
```

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить у преподавателя номер варианта задания.
2. В соответствии с номером вырианта выполнить задания лабораторной работы № 1, используя функции.
3. Оформить отчет о работе.

4. СОДЕРЖАНИЕ ОТЧЕТА

1. Ф.И.О. студента; № варианта;
2. Номер и название работы;
3. Цель работы;
4. Текст задания (постановка задачи);
5. Блок-схема алгоритма решения поставленной задачи;
6. Текст программы;
7. Результаты выполнения программы;
8. Выводы.

5. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Описание и объявление функции с параметрами и без параметров.
2. Оператор return. Использование в функциях.
3. Область действия и область видимости переменных.

4. Параметры и аргументы функций.
5. Передача параметров функции по значению и по ссылке.
6. Использование рекурсивных функций.
7. Функции с переменным числом параметров.
8. Использование указателей на функции.

Лабораторная работа 6.
Работа с файлами

1. ЦЕЛЬ РАБОТЫ

Получить начальные навыки написания программ на языке программирования С, использующих внешние файлы для получения и сохранения данных.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Доступ к файлам. Для работы с файлом необходимо открыть файл с помощью функции **fopen** из стандартной библиотеки **stdio.h.** Функция fopen берет внешнее имя, проводит некоторые обслуживающие действия и возвращает внутреннее имя ("указатель файла"), которое должно использоваться при последующих чтениях из файла или записях в него.

Указатель файла описывается следующим образом:

FILE *<идентификатор>;

Такая запись означает, что <идентификатор> является указателем на **FILE** (структура, описанная в **stdio.**h и содержащая информацию об открытом файле) и **fopen** возвращает указатель на **FILE.**

Например,

FILE *fd;

Обращение к функции **fopen** в программе имеет вид:

fd = fopen(name,mode);

где **name –** имя файла, которое задается в виде символьной строки, **mode –** режим открытия, который также является символьной строкой. Допустимыми режимами являются: чтение ("**r**"), запись ("**w**") и добавление ("**a**").

При открытии несуществующего файла он будет создан (если это возможно).

Открытие существующего файла на запись приводит к отбрасыванию его старого содержимого. Попытка чтения несуществующего файла является ошибкой. Ошибки могут быть обусловлены и другими причинами (например, попыткой чтения из файла, не имея на то разрешения). При наличии ошибки функция возвращает нулевое значение указателя **NULL.**

Функция fclose является обратной по отношению к **fopen;** она разрывает связь между указателем файла и внешним именем.

Основные функции для чтения представлены в табл. 4.

Таблица 4

Основные функции для чтения

Функция	Описание	Пример
getc	Возвращает следующий символ из файла	**c=getc(fd)**
putc	Помещает символ в файл	**putc(c,fd)**
fprintf	Форматированный ввод в файл	**fprintf(fd,"%d",i)**
fscanf	Форматированный вывод в файл	**fscanf(fd,"%d",&i)**
fgets	Читает символы из файла в строку	**fgets(s,n,fd)** – читает **n-1** символов в строку **s**
fputs	Помещает строку символов в файл	**fputs(s,fd)**

Ниже приведены примеры программ, использующие эти функции.

Пример 1. Запись строки в файл.

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

```
{       char s[]="Hello world";
        FILE *fd;
        fd=fopen("file_name.txt","w"); – создаем файл fale_name.txt
        if(fd==NULL)
        {       printf("\n Ошибка создания файла");
        return;
        }
        fputs(s,fd);
        fclose(fd);
        return;
}
```

Пример 2. Вывод на экран содержимого файла.

```
#include <stdio.h>
main()
{       char c;
        FILE *fd;
        fd=fopen("file_name.txt","r"); – открываем файл fale_name.txt
        if(fd==NULL)
        {       printf("\n Ошибка открытия файла");
        return;
        }
        c=getc(fd);
        while(c!=EOF) – сравниваем с с константой – конец файла
        {       printf("%c",c);
        c=getc(fd);
        }
        fclose(fd);
        return;
}
```

Произвольный доступ к файлам. Функции **fseek** позволяет передвигаться по файлу, не производя фактического чтения или записи. Функция имеет следующий синтаксис:

int fseek (FILE *fd, long offset, ing origin);

В этом случае текущая позиция в файле с дескриптором **fd** смещается на величину **offset** относительно места, заданного аргументом **origin: 0 (SEEK_SET)** – от начала, 1 **(SEEK_CUR**) – от текущей позиции, 2 **(SEEK_END)** – от конца файла. Последующее чтение или запись будут начинаться с новой позиции.

Пример 3. Функция чтения любого количества символов, начиная с произвольного места в файле.

```
#include <stdio.h>
int get (FILE *fd, long pos, int n)
{       char *s;
        if (fseek(fd,pos,0)>=0) – установка позиции
        return (fgets(s,n+1,fp));
        else
        return (-1);
}
```

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить у преподавателя номер варианта задания.
2. Составить алгоритм решения заданий № 1, 2.
3. Написать программы на языке С, реализующие алгоритмы п. 2.
4. Оформить отчет о работе.

4. ВАРИАНТЫ ЗАДАНИЙ

1. Произвести слияние двух текстовых файлов. Имена файлов вводятся с клавиатуры.
2. Разработать программу создания, дополнения и вывода на экран файла, содержащего текстовую информацию.
3. Разработать программу управления матрицами (добавление и удаление строк или столбцов). Матрица хранится в файле.
4. Разработать программу удаления в тексте, содержащемся в файле, лишних пробелов.
5. Разработать программу удаления в тексте, содержащемся в файле, всех пробелов.

6. Напишите программу, которая выводит на экран содержимое текстового файла.

7. Напишите программу, которая вычисляет среднее арифметическое чисел, находящихся в файле.

8. Напишите программу, которая позволяет просматривать текстовые файлы (выводит на экран содержимое файла), например, файлы исходных программ C++.

9. Напишите программу, которая дописывает в находящийся на диске файл имя, фамилию и номер телефона, например, вашего товарища. Если файла на диске нет, то программа должна создать его. В файле каждый элемент данных (имя, фамилия, телефон) должен находиться в отдельной строке.

10. Напишите программу, которая дописывает в находящийся на диске файл имя, фамилию и номер телефона, например, вашего товарища. Если файла на диске нет, то программа должна создать его. В файле все записи должны находиться последовательно в одной строке.

11. Выбросить из текста, находящегося в файле, заданный знак, где бы он не встречался.

12. Удалить из текста, находящегося в файле, каждое четное предложение.

13. В тексте перед каждым предложением, в котором встречается заданное слово, поставить восклицательный знак "!"

14. Отформатировать текст, находящийся в файле, следующим образом: каждое предложение должно иметь свой порядковый номер; начинаться с красной строки.

15. Определить, равны ли два заданных файла.

16. Заменить заданное слово в тексте, находящемся в заданном файле, на другое слово.

5. СОДЕРЖАНИЕ ОТЧЕТА

1. Ф.И.О. студента; № варианта;
2. Номер и название работы;
3. Цель работы;
4. Текст задания (постановка задачи);
5. Блок-схема алгоритма решения поставленной задачи;
6. Текст программы;

7. Результаты выполнения программы;
8. Выводы.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Доступ к файлам.
2. Открытие существующего файла на запись. Ошибки.
3. Основные функции для чтения.
4. Вывод на экран содержимого файла.
5. Произвольный доступ к файлам.

Лабораторная работа 7.
Работа со строками символов

1. ЦЕЛЬ РАБОТЫ

Получить начальные навыки написания программ на языке программирования С, работающих со строками символов.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Чаще всего строки символов описываются как массивы:

char a[40], *b;

В переменной **a** можно хранить строки с максимальной длиной 39 символов. Один байт в строках символов отдается под нулевой символ. Переменная **b** является указателем на строку символов:

b="Кафедра ИиАПС";

Основные функции для работы со строками и описанные в библиотеке **string.h**:

1. s**trcpy** - функция копирует строку **source** в строку **target**. Функция предполагает, что целевая строка имеет размер, достаточный для того, чтобы вместить содержимое строки-источника. Прототип функции **strcpy**:

char* strcpy(char *target, const char *source);

Пример 1:

```
char name[41] ;
strcpy(name, "Borland C++ 5");
```

2. **strdup - к**опирует одну строку в другую, при этом отводит необходимое количество памяти для целевой строки. Прототип функции **strdup**:

char* strdup(const char *source);

Функция копирует строку **source** и возвращает указатель на строку-копию.

Пример 2:

```
char *string1 = "Монархия в Испании";
char *string2;
string2 = strdup(string1);
```

3. **strncpy -** копирует заданное количество символов из одной строки в другую. Функция копирует **num** символов из строки **source** в строку **targe**t. Функция не выполняет ни усечение, ни заполнение строки. Прототип функции **strncpy**:

char * strncpy(char *target, const char *source, size_t num);

Пример 3:

```
char str1[] = "Pascal";
char str2[] = "Hello there";
strcnpy(strl, str2, 5);
```

4. **strlen -** возвращает количество символов в строке, в которое не включается нуль-терминатор. Прототип функции **strlen** таков:

size_t strlen (const char *string);

Функция **strlen** возвращает длину строки **string. size_t** — это имя, приписанное типу **unsigned int** оператором **typedef**.

Пример 4:

```
char str[] = "1234567890";
```

```
size_t i;
i = strlen(str);
```

5. **strcat -** функция добавляет к содержимому целевой строки содержимое строки-источника и возвращает указатель на целевую строку. Функция предполагает, что целевая строка может вместить содержимое объединенной строки. Прототип функции **strcat** таков:

char *strcat(char *target, const char *source);

Пример 5:

```
char string[81] ;
strcpy(string, "Turbo");
strcat (string, " C++");
```

6. **strncat** - добавляет к содержимому целевой строки указанное количество символов из строки-источника. Функция добавляет к содержимому целевой строки **num** символов из строки-источника и возвращает указатель на целевую строку. Прототип функции **strncat**:

char *strncat(char *target, const char *source, size_t num);

Пример 6:

```
char strl[81] = "Hello I am ";
char str2[41] = "Ivan Petrov";
strncat(strl, str2, 5);
```

7. **strcmp -** выполняет сравнение двух строк с учетом регистра символов. Прототип функции **strcmp**:

int strcmp(const char *strl, const char *str2);

Функция сравнивает строки **strl** и **str2**. Возвращает в качестве результата сравнения целую величину:

< 0 когда **strl** меньше, чем **str2**;

= 0 когда **strl** равна **str2**;

> 0 когда **strl** больше, чем **str2**.

Пример 7:

```
char stringl[] = "Borland C++";
char string2[] = "BORLAND C++";
i = strcmp(string1, string2);
```

8. **stricmp -** аналогична **strcmp**, но выполняет сравнение двух строк, не учитывая регистра символов.

9. **strncmp - а**налогична **strcmp**, но выполняет сравнение заданного количества символов двух строк с учетом регистра символов. Функция сравнивает первые **num** символов строк **strl** и **str2**. Прототип функции **strncmp**:

int strncmp(const char *strl, const char *str2, size_t num);

10. **strnicmp - а**налогична **stricmp**, но выполняет сравнение заданного количества символов двух строк без учета регистра символов. Функция сравнивает первые **num** символов строк **strl** и **str2**, не делая различия в регистре символов. Прототип функции **strnicmp**:

int strnicmp(const char *strl, const char *str2, size_t num);

11. **strlwr** - функция преобразует символы верхнего регистра в символы нижнего регистра в строке **source**. Другие символы не затрагиваются. Функция возвращает указатель на строку **source**. Прототип функции **strlwr**:

char* strlwr (char *source);

Пример 8:

```
char str[] = "HELLO THERE";
strlwr(str);
```

12. **strupr -** функция преобразует символы нижнего регистра в символы верхнего регистра в строке **source**. Другие символы не затрагиваются. Функция возвращает указатель на строку source. Прототип функции **strupr**:

char* strupr(char *source);

Пример 9:

```
char str[] = "Borland C++";
strupr(str);
```

13. **strrev** - функция обращает порядок символов в строке **str** и возвращает указатель на строку **str**. Прототип функции **strrev**:

char* strrev(char *str);

Пример 10:

```
char str[] = "Hello";
strrev(str) ;
```

14. **strchr -** Определяет первое вхождение символа в строку. Функция находит первое вхождение символа **c** в строку **target**. Функция возвращает указатель на символ в строке **target**, который соответствует заданному образцу **c**. Если символ **c** в строке не обнаруживается, функция возвращает 0. Прототип функции **strchr**:

char* strchr(const char *target, int c);

Пример 11:

```
char str[81] = "Borland C++";
char *strPtr;
strPtr = strchr(str, '+');
```

15. **strrchr - o**пределяет последнее вхождение символа в строке. Функция находит последнее вхождение символа c в строку **target**. Функция возвращает указатель на символ в строке **target**, который соответствует заданному образцу **c**. Если символ **c** в строке не обнаруживается, функция возвращает 0. Прототип функции **strrchr**:

char* strrchr(const char *target, int c);

Пример 12:

```
char str[81] = "Borland C++ is here";
char* strPtr;
```

strPtr = strrchr(str, '+');

16. **strspn - в**озвращает число символов с начала строки, совпадающих с любым символом из шаблона. Функция **strspn** возвращает число символов от начала строки **target**, совпадающих с любым символом из шаблона **pattern**. Прототип для функции **strspn**:

size_t strspn(const char *target, const char *pattern);

Пример 13:

```
char str[] = "Borland C++ 5";
char substr[] = "narlBod";
int index;
index = strspn(str, substr);
```

17. **strcspn -** просматривает строку и выдает число первых символов в строке, которые не содержатся в шаблоне. Функция **strcspn** просматривает строку **str1** и выдает длину подстроки, отсчитываемой с начала строки, символы которой полностью отсутствуют в строке **str2**. Прототип функции **strcspn**:

size_t strcspn(const char* str1, const char* str2);

Пример 14:

```
char strng[] = "The rain in Spain";
int i = strcspn(strng, " in");
```

18. **strpbrk -** просматривает строку и определяет первое вхождение любого символа из образца. Функция **strpbrk** ищет в строке **target** первое вхождение любого символа из образца **pattern**. Если символы из образца не содержатся в строке, функция возвращает 0. Прототип функции **strpbrk**:

char* strpbrk(const char* target, const char* pattern);

Пример 15:

```
char *str = "Hello there how are you";
char *substr = "hr";
```

```
char *ptr;
ptr = strpbrk(str, substr);
```

19. **strstr** - Функция ищет в строке **str** первое вхождение подстроки **substr**. Функция возвращает указатель на первый символ найденной в строке **str** подстроки **substr**. Если строка **substr** не обнаружена в строке **str**, функция возвращает 0. Прототип функции **strstr**:

```
char* strstr(const char *str, const char *substr);
```

Пример 16:

```
char str[] = "Hello there! how are you";
char substr[] = "how";
char *ptr;
ptr = strstr (str, substr);
```

20. **strtok -** Разбивает строку на подстроки на основании заданного набора символов-ограничителей. Подстроки иногда называются *лексемами.* Функция разбивает строку на лексемы, согласно символам-ограничителям, заданным в параметре **delimeters**. Функция **strtok** вводит символ **'\0'** после каждой лексемы. Прототип функции **strtok**:

```
char* strtok(char *target, const char * delimiters);
```

Пример 17:

```
#include <stdio.h>
#include <string.h>
int main()
{       char *str = "(Base_Cost + Profit) * Margin";
        char *tkn = "+*()";
        char *ptr = str;
        printf("%s\n", str);
        // Первый вызов функции
        ptr = strtok(str, tkn);
        printf("Лексемы этой строки: %s", ptr);
        while (ptr)
        {
```

```
        // Первый аргумент должен быть равен нулю
if ((ptr = strtok(0, tkn)) != 0)
        printf (",%s", ptr);
        }
        printf("\n");
        return 0;
}
```

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить у преподавателя номер варианта задания.
2. В соответствии с номером вырианта выполнить задания лабораторной работы № 1, используя функции.
3. Оформить отчет о работе.

4. ВАРИАНТЫ ЗАДАНИЙ

1. Написать программу, которая запрашивает имя пользователя и здоровается с ним.
2. Напишите программу, которая вычисляет длину введенной с клавиатуры строки.
3. Напишите программу, которая выводит на экран сообщение в "телеграфном" стиле: буквы сообщения должны появляться по одной, с некоторой задержкой.
4. Напишите программу, которая выводит код введенного пользователем символа. Программа должна завершать работу в результате ввода, например, точки.
5. Написать программу, которая выводит на экран первую часть таблицы кодировки символов (символы с кодами от 0 до 127). Таблица должна состоять из восьми колонок и шестнадцати строк. В первой колонке должны быть символы с кодом от 0 до 15, во второй – от 16 до 31 и т.д.
6. Написать программу, которая во введенной с клавиатуры строке преобразует строчные буквы русского алфавита в прописные.
7. Написать программу, которая удаляет из введенной с клавиатуры строки начальные пробелы.

8. Написать программу, которая проверяет, является ли введенная с клавиатуры строка целым числом.

9. Написать программу, которая проверяет, является ли введенная с клавиатуры строка двоичным числом.

10. Написать программу, которая проверяет, является ли введенная с клавиатуры строка шестнадцатеричным числом.

11. Написать программу, которая проверяет, является ли введенная с клавиатуры строка дробным числом.

12. Написать программу, которая преобразует введенное с клавиатуры восьмиразрядное двоичное число в десятичное.

13. Написать программу, которая преобразует введенное с клавиатуры двухразрядное шестнадцатеричное число в десятичное.

14. Написать программу, которая преобразует введенное пользователем десятичное число в число в указанной системе счисления (от 2 до 10).

15. Написать программу, которая преобразует введенное пользователем десятичное число в шестнадцатеричное.

16. Написать программу, которая вычисляет значение выражения $N0O1N1O2 \ldots OkNk$, где Ni – целое одноразрядное число; Oi – один из двух знаков простейших арифметических действий: сложения (+) или вычитания.

5. СОДЕРЖАНИЕ ОТЧЕТА

1. Ф.И.О. студента; № варианта;
2. Номер и название работы;
3. Цель работы;
4. Текст задания (постановка задачи);
5. Блок-схема алгоритма решения поставленной задачи;
6. Текст программы;
7. Результаты выполнения программы;
8. Выводы.

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как описываются строки символов?
2. Основные функции для работы со строками?

3. Как сравнить две строки, не учитывая регистра символов?
4. Функция преобразования символов верхнего регистра в символы нижнего регистра в строке?
5. Функция возвращения указателя на символ в строке.
6. Какая функция определяет последнее вхождение символа в строке?

Лабораторная работа 8.
Работа с графикой

1. ЦЕЛЬ РАБОТЫ

Получить начальные навыки написания программ на языке программирования С, работающих в графическом режиме.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Графический режим дисплея предназначен для вывода графиков, рисунков и т.п. В этом режиме можно выводить также и текстовую информацию.

В программах на языке С прежде вывода на экран графической информации необходимо произвести инициализацию графического режима, а по окончании – закрытие графического режима. Графический режим определяет разрешающую способность экрана и количество цветов и должен поддерживаться видеоадаптером. Функции, использующиеся для инициализации и деинициализации графического режима, а также для вывода информации на экран в графическом режиме, содержатся в заголовочном файле graphics.h.

Инициализация графического режима. Для этого используется функция **initgraph**, имеющая следующий синтаксис:

void far initgraph(int far *graphdriver, int far *graphmode, char far *pathtodriver)

Функция **initgraph** инициализирует графическую систему путем загрузки графического драйвера с диска и переводит систему в графический режим.

Параметры функции:

– ***pathodriver** – определяет маршрут, по которому initgraph будет искать графические драйверы (если строка пуста, то поиск осуществляется в текущем каталоге);

– ***graphdriver** – целое, которое определяет используемый графический драйвер;

– ***graphmode** – целое, которое определяет исходный графический режим (если ***graphdriver** не равен **DETECT**,

***graphmode** устанавливается в наивысшее разрешение, допустимое для данного драйвера).

Возвращаемое **initgraph** значение представляет собой код ошибки, который необходимо проанализировать, прежде чем приступать к работе в графическом режиме. В случае успешного завершения код равен 0.

Закрытие графического режима. Закрытие графического режима осуществляется при помощи функции closegraph, имеющей следующий синтаксис:

void far closegraph(void).

Функция **closegraph** освобождает всю память, выделенную под графическую систему, затем восстанавливает экран в режим, который был до вызова **initgraph**.

Пример 1. Программа, инициализирующая и деинициализирующая графический режим (в программе используется автоопределение режима):

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
int main(void)
{     int graphdriver = DETECT, gmode, errorcode;
      initgraph(&graphdriver,&gmode,"");
      errorcode = graphresult(); – получение результата инициализации
      if(errorcode != grOk) – если ошибка */
      {     printf("Ошибка
      :%s\n",grapherrormessage(errorcode));
      printf("Для останова нажмите любую клавишу\n");
```

```
        getch();
        return(1); – завершение с кодом ошибки
        }
        line(0,0,getmaxx(),getmaxy()); – построение диагональной линии
        getch();
        closegraph();
        return 0;
}
```

Таблица 5

Основные функции работы с графикой

Функция	Описание	Пример
line	Рисует линию между двумя точками (x1, y1) и (x2, y2)	**line(x1,y1,x2,y2)**
getmaxx и **getmaxy**	Возвращает максимальные значения x и y для текущего драйвера и режима	**x_max=getmaxx()** и **y_max=getmaxy()**
setcolor	Изменяет текущий цвет рисования	**setcolor(RED)** – установка красного цвета
setbkcolor	Изменяет текущий цвет фона	**setbkcolor(GREEN)**
outtextxy	Отображает строку текста в окне экрана в заданной позиции (x, y)	**outtextxy(x,y,"Hello")**
putpixel	Отображает точку цвета color и с координатами (x, y)	**putpixel(x,y,color)**
bar	Рисует двухмерный заполненный прямоугольник (координаты в пикселях)	**bar(left,top,right,bottom)** **right, bottom** – правый нижний **left,top** – левый верхний угол
bar3d	Рисует трехмерный закрашенный прямоугольный столбец	**bar3d(left,top,right,bottom,depth,topflag)** **depth** – глубина прямоугольника **topflag** – определяет, будет ли рисоваться вершина столбца (topflag=0, вершина не

		рисуется, иначе рисуется)
cleardevice	Стирает весь графический экран и переносит текущую позицию в начало экрана	**cleardevice**()

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить у преподавателя номер варианта задания (см. табл. 2).

2. Разработать программу построения графика функции, заданной в соответствии с вариантом. Границы исследуемого участка функции (min *x* и max *x*), масштаб по осям *OX* и *OY* задаются пользователем с клавиатуры. Предусмотреть вывод координатных осей на экран и их разметку в соответствии с заданными масштабом и границами

3. Оформить отчет о работе.

4. СОДЕРЖАНИЕ ОТЧЕТА

1. Ф.И.О. студента; № варианта.
2. Номер и название работы.
3. Цель работы.
4. Текст задания (постановка задачи).
5. Блок-схема алгоритма решения поставленной задачи.
6. Текст программы.
7. Результаты выполнения программы.
8. Выводы.

5. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Инициализация графического режима.
2. Закрытие графического режима.
3. Основные функции работы с графикой.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Березин, Б. И. Начальный курс С и С++ / Б. И. Березин, С. Б. Березин. – М.: Диалог – МИФИ, 1996. – 288 с.

2. Громов, Ю. Ю. Программирование на языке СИ: учеб. пособие / Ю. Ю. Громов, С. И. Татаренко. – Тамбов, 1995. – 169 с.

3. Жешке, Р. Толковый словарь стандарта языка Си / Р. Жешке. – СПб.: Питер, 1994.

4. Информатика: задачник-практикум: в 2 т. / под ред. И. Г. Семакина, Е. К. Хеннера. – М.: Бином; Лаборатория Знаний, 2002. – 304 с.

5. Касаткин, А. И. Профессиональное программирование на языке С. От Turbo C к Borland C++ / А. И. Касаткин. – Минск : Высшая школа, 1995.

6. Керниган, Б. Язык программирования С / Б. Керниган, Д. Ритчи. – 2-е изд. – М.: Финансы и статистика, 1992.

7. Собоцинский, В. В. Практический курс Turbo C++ / В. В. Собоцинский. – М.: Свет, 1993.

СОДЕРЖАНИЕ

Составитель
Полетаев Вадим Алексеевич

ПРОГРАММИРОВАНИЕ И ОСНОВЫ АЛГОРИТМИЗАЦИИ

Методические указания к лабораторным работам для студентов специальности 220301 «Автоматизация технологических процессов и производств (в машиностроении)»

Печатается в авторской редакции

Подписано в печать 16.03.2011. Формат 60×84/16.
Бумага офсетная. Отпечатано на ризографе. Уч.-изд. л. 2,2
Тираж 30 экз. Заказ
ГУ КузГТУ. 650000, Кемерово, ул. Весенняя, 28.
Типография ГУ КузГТУ. 650000, Кемерово, ул. Д.Бедного, 4 А.