

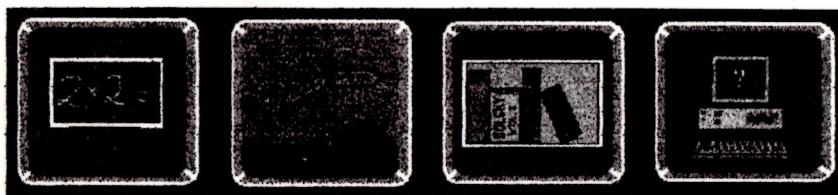
МИНИСТЕРСТВО ОБРАЗОВАНИЯ УКРАИНЫ
ДОНЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ И ЗАДАНИЯ
К ЛАБОРАТОРНЫМ РАБОТАМ ПО КУРСУ
“ОСНОВЫ ПРОГРАММИРОВАНИЯ
И АЛГОРИТМИЧЕСКИЕ ЯЗЫКИ”

(для студентов специальности “Программное обеспечение”
и “ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ”)

(для студентов специальности “Информационные системы в
менеджменте”)

Часть 1



ДОНЕЦК ДонГТУ 1997

Таблица 1.2 -Перечень используемого программного обеспечения

N п/п	Наименование программы	Назначение программы	Версия
1	DOS-HELP	Обучающая программа по командам MS-DOS	1.0
2	Visual Bloks	Обучающая программа по этапу алгоритмизации (CASE-система)	1.0
3	Chainik	Обучающая программа по языку Си	1.0
4	Turbo-C	Компилятор языка программирования Си	2.0, 3.0
5	Turbo-C++	Компилятор языка программирования Си++	1.0
6	Borland-C++	Компилятор языка программирования Си++	2.0, 3.x
7	Turbo-Pascal	Компилятор языка программирования Паскаль	5.0, 7.0
8	Borland-Pascal	Компилятор языка программирования Паскаль	7.0

2 ЛАБОРАТОРНАЯ РАБОТА N 1. КАЛЕНДАРНЫЕ ЗАДАЧИ НА ЯЗЫКЕ СИ

Цель работы: освоить приемы алгоритмизации при организации разветвляющихся и повторяющихся вычислений (на примере решения календарных задач) при использовании целочисленной арифметики; приемы ввода исходных данных и проверки их ограничений; приемы инициализации исходных данных (скалярных и структурированных); приемы формирования и вывода результатов, количество которых формируется в программе динамически.

Контрольные вопросы:

- 1) Как организуются разветвляющиеся алгоритмы ?
- 2) Как организуются повторяющиеся алгоритмы ?
- 3) Как выполняется описание в языке Си скалярных данных и структурированных данных типа "массив"?
- 4) Как выполняется инициализация данных в языке Си ?
- 5) В чем заключается различие в семантике циклов с неизвестным числом повторений (с пред - и постусловием), используемых в языке Си ?
- 6) Почему массив, содержащий значения количества дней во всех месяцах года, нельзя объявить в программе на языке Си с квалификатором *const* ?

2.1 Приемы алгоритмизации и программирования при решении календарных задач

При решении календарных задач достаточным является использование целочисленной арифметики. Вся информация о предметной области представляется данными целого типа:

- *день, месяц, год* - своими порядковыми номерами;
- *дни недели* - своими порядковыми номерами в пределах недели (например, 1 - понедельник, 2 - вторник, ... 7 - воскресенье).

2.1.1 Ввод исходных данных и проверка ограничений

При вводе исходных данных проверка ограничений на их корректность может быть организована различными способами.

В первом случае при нарушении ограничений пользователю сообщается о его ошибке; дальнейшее выполнение программы прекращается. Например,

```
int dn; /* порядковый номер дня недели*/
printf("Введите день недели текущего дня (1..7):\n");
scanf("%d", &dn);
if (dn>7 || dn<1)
    {printf("Вы ошиблись при вводе !!!");
     exit (-1); /* завершение выполнения программы*/
    }
```

Во втором случае при нарушении ограничений пользователю сообщается о его ошибке и предлагается повторить ввод этих исходных данных; дальнейшее выполнение программы продолжается. В этом случае следует использовать циклы с неизвестным числом повторений.

Вариант 1. Использование цикла с предусловием. Например:

```
int dn; /* порядковый номер дня недели*/
printf("Введите день недели текущего дня (1..7):\n");
scanf("%d", &dn);
while (dn>7 || dn<1)
    {printf("Вы ошиблись при вводе !!! ");
     printf("Повторите ввод дня недели текущего дня (1..7):\n");
     scanf("%d", &dn);
    }
```

Вариант 2. Использование цикла с постусловием. Например:

```
int dn; /* порядковый номер дня недели*/
int fl = 0; /* признак: в данных нет ошибки */
do
{ if (fl)
    {printf("Вы ошиблись при вводе !!!\n ");
     printf("Введите день недели текущего дня (1..7):\n");
     scanf("%d", &dn);
     if (dn>7 || dn<1)
         fl = 1; /* признак: в данных ошибка */
    }
while (dn>7 || dn<1);
```

При вводе исходных данных этой предметной области проверка ограничений имеет свои особенности.

1) Проверку правильности введенной даты следует начинать с года. После проверки ограничений для года можно установить признак високосности года, который затем потребуется для установки значения количества дней в месяце феврале. Год считается високосным, если он кратен 4, но не кратен 100, или кратен 400.

2) Затем выполняется проверка правильности значения месяца года. В случае високосности года устанавливается значение количества дней в месяце феврале с учетом этого признака.

3) В заключение выполняется проверка правильности значения для месяца с учетом количества дней в данном месяце.

2.1.2 Инициализация исходных данных

Часть исходных данных не вводится пользователем, они известны заранее. Это, например, такие данные, как количество дней в месяцах года или даты праздничных дней государства.

Пользователя следует освободить от рутинной работы по вводу этих данных. В языке Си эта задача решается с помощью средств инициализации данных при их декларации в программе. Например, инициализация массива, содержащего значения количества дней во всех месяцах года:

```
#define NN 12
/* массив количества дней в месяцах года */
int kdm[NN]=
    {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

2.1.3 Выбор структур данных для представления информации предметной области

Если в задаче необходимо иметь список некоторых дат, то для представления этой информации достаточно воспользоваться структурой данных типа "массив". Один из возможных способов представления такого списка:

```
#define KPD 11
int mas_pr[2][KPD]={ {1, 2, 7, 8, 1, 2, 9, 28, 25, 7, 8},
                     {1, 1, 1, 3, 5, 5, 5, 6, 8, 11, 11} };
```

Здесь KRD - это количество государственных праздничных дней Украины, исключая праздники, привязанные к православному церковному календарю и изменяющие свою дату год от года (например, Пасха, Троица).

Первое измерение этого массива, моделирующего список дат, равно 2: первая строка соответствует дню даты, вторая - месяцу даты. Второе измерение массива соответствует количеству элементов в списке дат.

Так, пара элементов `mas_pr[0][0]` и `mas_pr[1][0]` моделирует день первого января, а `mas_pr[0][3]` и `mas_pr[1][3]` - 8 Марта.

Элементы массива mas_pr могут быть постоянными в программе, тогда в декларации этого массива можно добавить квалификатор const, чтобы пользователь не изменял значения его элементов в программе.

2.1.4 Формирование результатов, количество которых заранее неизвестно

Если в задаче необходимо сформировать список некоторых дат, то для представления этой информации достаточно воспользоваться структурой данных типа "массив". Один из возможных способов представления такого списка:

```
#define KRD 304  
int mas_rd[2][KRD];
```

Здесь KRD - это максимально возможное количество рабочих дней в году.

Так как реальное количество сформированных элементов списка дат заранее неизвестно, то для хранения значения количества реально заполненных элементов массива необходимо иметь дополнительную переменную, "сопровождающую" этот массив (так называемый "указатель занятости массива"). Первоначальное значение этой переменной должно быть таковым, чтобы сигнализировать об отсутствии элементов в списке (моделирование ситуации "список дат пустой").

Здесь возможны два варианта семантики "указателя занятости массива".

Первый вариант соответствует ситуации, когда "указатель занятости" указывает на последний занятый элемент в массиве.

Так как младший индекс элемента массива в языке Си равен 0, то начальное значение "указателя занятости" может быть меньше 0:

```
#define KRD 304  
int mas_rd[2][KRD]; /* формируемый список дат рабочих дней*  
/* указатель занятости mas_rd - "последний занятый" */  
int uk_mas_rd = -1;
```

Второй, вариант соответствует ситуации, когда "указатель занятости" указывает на первый свободный элемент в массиве. Так как младший индекс элемента массива в языке Си равен 0, то начальное значение "указателя занятости" будет равно 0:

```
/* указатель занятости mas_rd - "первый свободный" */  
int uk_mas_rd = 0;
```

При формировании списка дат запись информации в этот массив производится в первый "свободный" (еще не заполненный) элемент массива в соответствии в семантикой "указателя занятости".

1) "Указатель занятости" - последний занятый элемент в массиве. В этом случае необходимо сначала переместить "указатель занятости" на один элемент массива вперед, а затем произвести запись информации в массив по этому индексу:

```
uk_mas_rd++; /* перемещение на "первый свободный"*/  
/* ... формируем значение дня даты рабочего дня в переменной d  
и значение месяца даты рабочего дня в переменной m*/  
/* записываем в массив mas_rd: */  
mas_rd[0][uk_mas_rd]=d;  
mas_rd[1][uk_mas_rd]=m;
```

После выполнения этих операций дата записана в элемент массива `mas_rd` с номером `uk_mas_rd`, причем он снова указывает на "последний занятый".

2) "Указатель занятости" - первый свободный элемент в массиве. В этом случае необходимо сначала произвести запись информации в массив по этому индексу, а затем переместить "указатель занятости" на один элемент массива вперед:

```
/* ... формируем значение дня даты рабочего дня в переменной d  
и значение месяца даты рабочего дня в переменной m*/  
/* записываем в массив mas_rd: в "первый свободный" элемент */
```

```

mas_rd[0][uk_mas_rd]=d;      mas_rd[1][uk_mas_rd]=m;
uk_mas_rd++; /* перемещение на "первый свободный"*/

```

После выполнения этих операций дата записана в элемент массива *mas_rd* с номером *uk_mas_rd*, причем он снова указывает на "первый свободный".

При изменении значения "указателя занятости" необходимо контролировать ситуацию "переполнения" списка дат путем сравнения значения "указателя занятости" с предельным значением:

```

uk_mas_rd++; /* перемещение перед записью в массив*/
if(uk_mas_rd >= KRD)
{printf ("Внимание: список уже переполнен!!!\n");
 printf ("Последняя запись в массив не выполнена!!!\n");
 exit (-1);
}

```

В этой ситуации добавлять элементы в массив уже нельзя.

2.1.5 Вывод результатов, количество которых заранее неизвестно

Вывод таких результатов следует организовать с помощью цикла с известным числом повторений, используя значение "указателя занятости" сформированного массива. Выводу значений также должна предшествовать проверка на ситуацию "сформированный список пустой".

Здесь также необходимо учитывать семантику этого указателя.

1) "Указатель занятости" - последний занятый элемент в массиве. В этом случае предельное значение изменения переменной цикла должно быть строго равно значению "указателя занятости":

```

#define KRD 304
int mas_rd[2][KRD]; /* формируемый список дат рабочих дней*/
/* указатель занятости mas_rd - "последний занятый" */
int uk_mas_rd = -1;
int i; /* промежуточная переменная для вывода массива */
...
/* вывод сформированного списка дат рабочих дней */

```

```

if(uk_mas_rd == -1)
    printf ("Список дат рабочих дней пуст !!!:\n");
else
{
    printf ("Список дат рабочих дней:\n");
    for (i=0; i<=uk_mas_rd; i++)
        printf ("%d-я дата: (%d. %d)\n",
               i+1, mas_rd[0][i], mas_rd[1][i]);
}

```

2) "Указатель занятости" - первый свободный элемент в массиве. В этом случае предельное значение изменения переменной цикла должно быть строго меньше значения "указателя занятости":

```

#define KRD 304
int mas_rd[2][KRD]; /* формируемый список дат рабочих дней*/
/* указатель занятости mas_rd - "первый свободный" */
int uk_mas_rd = 0;
int i; /* промежуточная переменная для вывода массива */
...
/* вывод сформированного списка дат рабочих дней */
if(uk_mas_rd == 0)
    printf ("Список дат рабочих дней пуст !!!:\n");
else
{
    printf ("Список дат рабочих дней:\n");
    for (i=0; i< uk_mas_rd; i++)
        printf ("%d-я дата: (%d. %d)\n",
               i+1, mas_rd[0][i], mas_rd[1][i]);
}

```

2.1.6 Организация ветвящихся алгоритмов при вычислении значения признаков

В языке Си отсутствует тип данных логический. Для его моделирования используется тип данных int. Целочисленный 0

соответствует значению "ложь"; целое, отличное от 0 - значению "истина".

Для формирования значения логического признака (наличие некоторого свойства или его отсутствие) достаточно организовать алгоритм, ветвящийся на два направления.

Возможны три способа формирования значения логического признака. Рассмотрим их на примере формирования значения признака высокосности года.

1) С использованием инструкции IF_ELS:

```
int v_g;           /* признак высокосности года */
if(g % 4 == 0 && g % 100 != 0 || g % 400 == 0)
    v_g=1;
else
    v_g=0;
```

2) С использованием оператора "?":

```
v_g = (g % 4 == 0 && g % 100 != 0 || g % 400 == 0)?1:0;
```

3) С использованием типа результата и свойств логических операторов:

```
v_g = g % 4 == 0 && g % 100 != 0 || g % 400 == 0;
```

2.1.7 Организация связи между датой года и порядковым днем недели, соответствующим этой дате

"Вечный календарь" [2]. Даны натуральные числа a , b , c , которые обозначают число, месяц, год. Определить день недели, на который падает указанная дата. При решении этой и некоторых подобных задач можно считать, что исследуемая дата лежит в диапазоне от 1582 до 4902 гг. Как установлено, в этом случае номер дня недели (воскресенье имеет номер 0, понедельник - 1, ..., суббота - 6) равен остатку от деления на 7 значения выражения

$$[2.6*m-0.2]+d+y+[y/4]+[c/4]-2*c, \quad (1.1)$$

где d - номер дня в месяце (1, 2, ..);

m - номер месяца в году, нумерация начинается с марта (март имеет номер 1, апрель - 2, ..., декабрь - 10, январь и февраль считаются месяцами с номерами 11 и 12 предыдущего года);

у - две младшие цифры года (00, ..., 99);
с - две старшие цифры года (15, ..., 49);
[x] означает целую часть числа x.

Для создания более универсального календаря, охватывающего все годы, можно использовать непосредственный подсчет, основанный на том, что 1 января 1 года нашей эры было понедельником.

2.2 Пример

Определить дату дня, следующего за заданным (d , m , g).

2.2.1 Постановка задачи

2.2.1.1 Исходные данные

d ,	{ день даты текущего дня }
m ,	{ месяц даты текущего дня }
g : целые;	{ год даты текущего дня }

2.2.1.2 Ограничения

$g > 0$;	{ годы нашей эры }
$1 \leq m \leq 12$;	{ месяцы года }
$1 \leq d \leq 31$;	{ дни месяца }

2.2.1.3 Результаты

d_1 ,	{ день даты следующего дня }
m_1 ,	{ месяц даты следующего дня }
g_1 : целые;	{ год даты следующего дня }
сообщ: строка[1..60];	{ сообщение об ошибках }

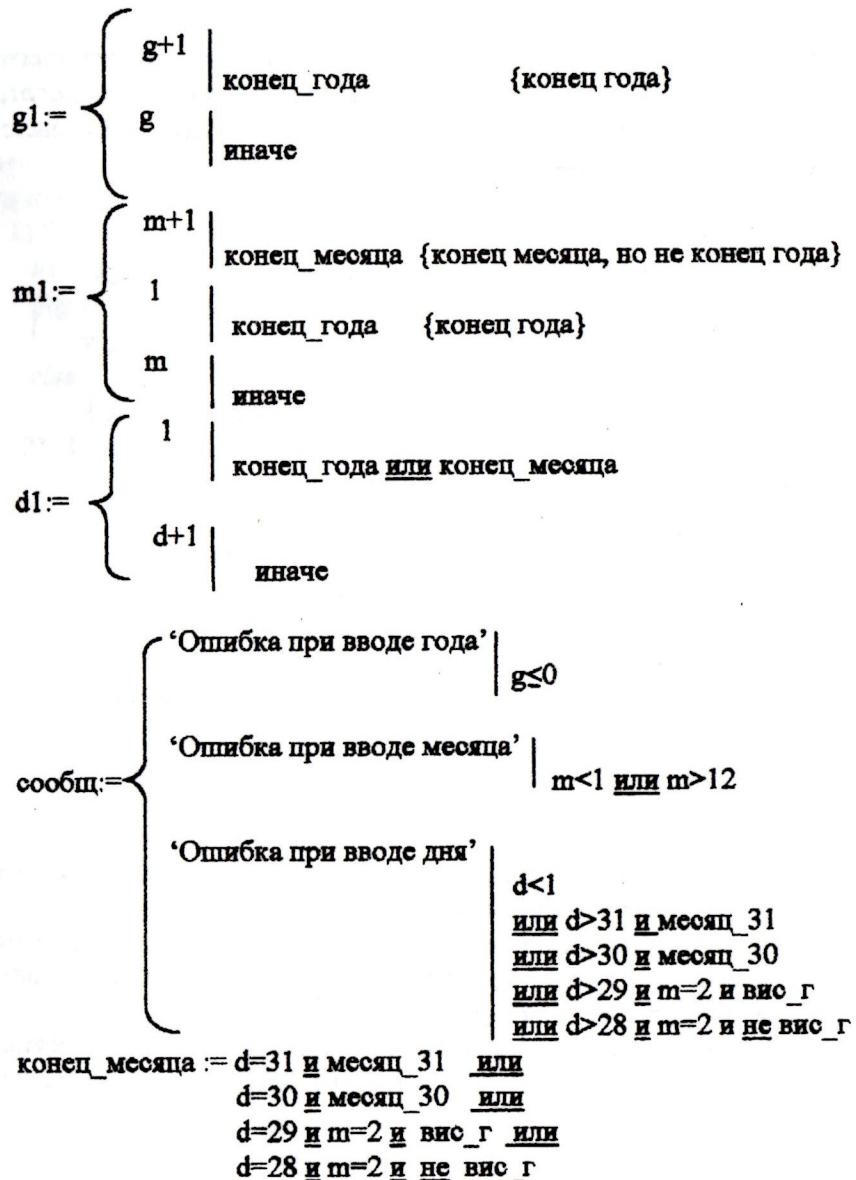
2.2.1.4 Связь

Если текущая дата есть последний день года, то переходим на начало следующего года: $d_1=1$; $m_1=1$; $g_1=g+1$.

Если текущая дата есть последний день месяца (кроме последнего дня года), то переходим на начало следующего месяца: $d_1=1$; $m_1=m+1$; $g_1=g$.

В остальных случаях (текущая дата в пределах месяца) переходим к следующему дню текущего месяца: $d_1=d+1$; $m_1=m$; $g_1=g$.

2.2.2. Метод решения



месяц_30 := m=4 или m=6 или m=9 или m=11
 месяц_31 := m=1 или m=3 или m=5 или m=7 или m=8 или m=10
 конец_года := d=31 и m=12
 вис_г := g кратен 4 и не (g кратен 100) или g кратен 400
 Промежуточные данные:
 вис_г, {признак високосного года}
 конец_года, {признак конца года}
 конец_месяца, {признак конца месяца}
 месяц_31, {признак месяца из 31 дня}
 месяц_30: логические; {признак месяца из 30 дней}

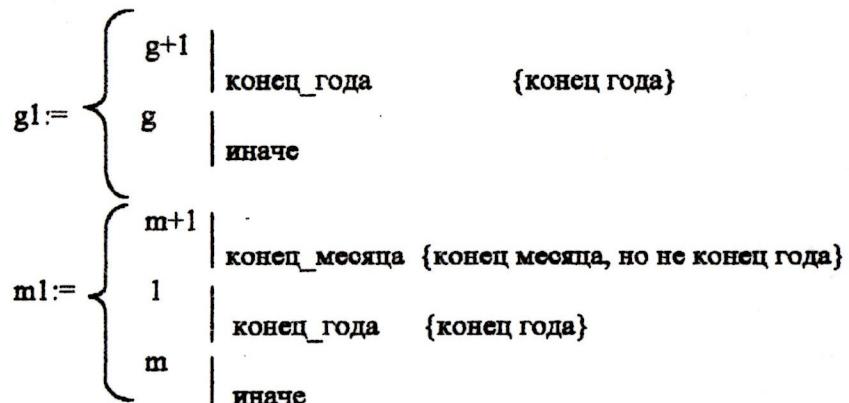
Примечание: Этот метод можно записать короче, если ввести вспомогательный массив, содержащий значения количества дней во всех месяцах года:

kdm [1..12]

31	28	31	30	31	30	31	31	30	31	30	31
1	2	3	4	5	6	7	8	9	10	11	12

kdm₁ соответствует количеству дней в первом месяце года (январе), а kdm₁₂ - в последнем месяце (декабре).

С использованием этого массива метод решения рассматриваемой задачи выглядит так:



$d1 := \begin{cases} 1 & | \text{конец_года или конец_месяца} \\ d+1 & | \text{иначе} \end{cases}$

$\text{сообщ:} = \begin{cases} \text{'Ошибка при вводе года'} & | g \leq 0 \\ \text{'Ошибка при вводе месяца'} & | m < 1 \text{ или } m > 12 \\ \text{'Ошибка при вводе дня'} & | d < 1 \\ & \text{или } d > kdm_m \end{cases}$

конец_месяца := $d = kdm_m$ и $m \neq 12$

$kdm_2 := kdm_2 + 1$

| вис_г

конец_года := $d = 31$ и $m = 12$

вис_г := g кратен 4 и не (g кратен 100) или g кратен 400

$kdm_{12} := 31$

$kdm_{11} := 30$

$kdm_{10} := 31$

$kdm_9 := 30$

$kdm_8 := 31$

$kdm_7 := 31$

$kdm_6 := 30$

$kdm_5 := 31$

$kdm_4 := 30$

$kdm_3 := 31$

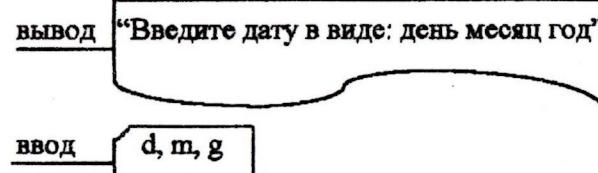
$kdm_2 := 28$

$kdm_1 := 31$

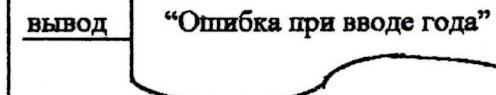
Промежуточные данные:
 вис_г, {признак високосного года}
 конец_года, {признак конца года}
 конец_месяца: логические; {признак конца месяца}
 kdm : массив [1..12] целых; {количество дней в месяцах года}

2.2.3 Алгоритм

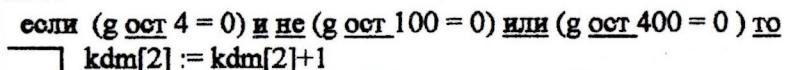
$kdm[1] := 31$
 $kdm[2] := 28$
 $kdm[3] := 31$
 $kdm[4] := 30$
 $kdm[5] := 31$
 $kdm[6] := 30$
 $kdm[7] := 31$
 $kdm[8] := 31$
 $kdm[9] := 30$
 $kdm[10] := 31$
 $kdm[11] := 30$
 $kdm[12] := 31$



если $g \leq 0$



иначе



```

если m<1 или m>12
    вывод "Ошибка при вводе месяца"
иначе
    если d<1 или d>kdm[m]
        вывод "Ошибка при вводе дня"
иначе
    конец_года := d = 31 и m = 12
    конец_месяца := d = kdm [m] и m ≠ 12
    если конец_года
        g1 := g + 1
    иначе
        g1 := g
    если конец_месяца
        m1 := m + 1
    иначе
        если конец_года
            m1 := 1
        иначе
            m1 := m
    если конец_года или конец_месяца
        d1 := 1
    иначе
        d1 := d + 1

```

```

вывод "Дата дня, следующего за",
d, ".", m, ".", g, "есть",
d1, ".", m1, ".", g1

```

Фрагмент алгоритма, вычисляющий значения результатов $d1, m1, g1$ может быть записан иначе:

```

если конец_года
    g1 := g + 1
    m1 := 1
    d1 := 1
иначе
    если конец_месяца
        g1 := g
        m1 := m + 1
        d1 := 1
    иначе
        g1 := g
        m1 := m
        d1 := d + 1

```

2.2.4 Таблица трассировки

Таблица трассировки алгоритма решения этого примера приведена в табл. 2.1.

Тесты, приведенные в ней, соответствуют следующим ситуациям:

- 1) ошибки при вводе даты (соответственно: года, месяца и дня);
- 2) возможные варианты результатов (переход к новому году; переход к новому месяцу; следующая дата находится в пределах текущего месяца)

Таблица 2.1 Таблица трассировки

Исходные данные		Промежуточные данные						Условия						Результаты				
d	m	g	v_g	end_y	end_m	end_l	kdm	g<=	g>4	g%	m<1	m>12	d<1	d> kdm _{m-1}	d1	m1	g1	сообщ
12	5	0			1			=0	=0	400								Ошибка при вводе года
3	0	1980				0	1	1	0									Ошибка при вводе месяца
25	13	1977		0		28					0	1						Ошибка при вводе дня
0	7	1990			0	28					0	0	1					Ошибка при вводе месяца
31	4	1990			0	28					0	0	0					Ошибка при вводе дня
											0	0	0					Ошибка при вводе месяца

Окончание табл 2.1

d	m	g	v_g	end_y	end_m	end_l	kdm	g<=	g>4	g%	m<1	m>12	d<1	d> kdm _{m-1}	d1	m1	g1	сообщ
31	12	1997		0		28		=0	=0	400								Ошибка при вводе года
				0	1	0					0	0	0					Ошибка при вводе месяца
					1	0												Ошибка при вводе дня
30	4	1990		0		28					0	0	0					Ошибка при вводе месяца
				0	0	1												Ошибка при вводе дня
29	2	1996		1		29					0	1	0					Ошибка при вводе года
				0	0	1												Ошибка при вводе месяца
					1	0												Ошибка при вводе дня
12	4	1961		0		28					0	0	0					Ошибка при вводе месяца
				0	0	0												Ошибка при вводе дня

2.2.5 Текст программы на языке Си

```

/* ===== КАЛЕНДАРНЫЕ ЗАДАЧИ ===== */
/*      ЛАБОРАТОРНАЯ РАБОТА N 1      */
/*    студентки гр. МИ93 Ивановой С.   */
/* Найти дату дня, следующего за данным (d.m.g.) */
#include <stdio.h>
#define NN 12
main()
{
/*===== инструкции описания (декларации) =====*/
/*===== ИСХОДНЫЕ ДАННЫЕ =====*/
    int d, /* день исходной даты */
        m, /* месяц исходной даты */
        g; /* год исходной даты */
/*===== РЕЗУЛЬТАТЫ =====*/
    int d1, /* день искомой даты */
        m1, /* месяц искомой даты */
        g1; /* год искомой даты */
/*===== ПРОМЕЖУТОЧНЫЕ ДАННЫЕ =====*/
    int end_m, /* признак конца месяца */
        end_y, /* признак конца года */
        v_g; /* признак высокосности года */
/* массив количества дней в месяцах года */
    int kdm [NN]=
        {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
/*===== ПРОВЕРКА ОГРАНИЧЕНИЙ =====*/
    printf("Введите исходную дату в виде: день месяц год\n");
    scanf("%d%d%d", &d, &m, &g);
/*===== ограничения на год =====*/
    while(g<=0)
    {
        printf("Вы ошиблись. Повторите ввод года.\n");
        scanf("%d", &g);
    }
/*===== формирование признака высокосности года =====*/
    v_g = (g % 4 == 0 && g % 100 != 0 || g % 400 == 0)?1:0;
}

```

```

/* ----корректировка количества дней в феврале----*/
/* 1-й вариант: с использованием IF_ELSE */
if(v_g) kdm[1]++;
/* 2-й вариант: без использования IF_ELSE */
kdm[1] += v_g;
/* ----- ограничения на месяц -----*/
while(m<1 || m>12)
{
    printf("Вы ошиблись. Повторите ввод месяца.\n");
    scanf("%d", &m);
}
/* ----- ограничения на день -----*/
while(d<1 || d>kdm[m-1])
{
    printf("Вы ошиблись. Повторите ввод дня.\n");
    scanf("%d", &d);
}
/*===== ИСПОЛНИМАЯ ЧАСТЬ ПРОГРАММЫ =====*/
/* --- формирование признака КОНЕЦ_ГОДА --- */
end_y = d==31 && m==12;
/* --- формирование признака КОНЕЦ_МЕСЯЦА --- */
end_m = d==kdm[m-1] && m!=12;
/* --- формирование искомого года (g1) --- */
/* 1-й вариант: с использованием IF_ELSE */
if(end_y)
    g1=g+1;
else
    g1=g;
/* 2-й вариант: с использованием оператора "?" */
/* g1=(end_y)?g+1:g;*/
/* --- формирование искомого месяца (m1) --- */
if(end_y) /* конец года*/
    m1=1;
else
    if(end_m) /* конец месяца*/
        m1=m+1;
    else
        m1=m;
}

```

```

/* ---- формирование искомого дня (d1) ---- */
/* 1-й вариант: с использованием IF_ELS */
if(end_y || end_m) /* конец дня или месяца */
    d1=1;
else
    d1=d+1;
/* 2-й вариант: с использованием оператора "?" */
/* d1=(end_y || end_m)?1:d + 1; */
/* ===== ВЫВОД РЕЗУЛЬТАТОВ ===== */
printf("Дата, следующая за %d.%d.%d г. есть %d.%d.%d г.\n",
       d, m, g, d1, m1, g1);
getch();
/* для задержки выведенных результатов на экране */
/*      до нажатия любой клавиши      */
}

```

2.3 Варианты заданий

- Сколько дней осталось до конца текущего квартала от заданной даты?
- Отпуск работающего начинается в день d.m.g и длится k дней. Подсчитать, сколько воскресений приходится на отпуск.
- Учебная четверть в школе начинается в день d.m.g. и длится k дней. Подсчитать, сколько праздничных дней приходится на эту четверть.
- Сколько раз в заданном году конец месяца приходится на среду?
- Задана дата d.m.g. Сколько полных кварталов прошло от начала этого года?
- Человек родился в день d.m.g. Сколько лет было этому человеку, когда его очередной день рождения впервые пришелся на воскресенье?
- День Шахтера отмечается в последнее воскресенье августа. Определить число, на которое в заданном году приходится День Шахтера.

8. Врач ведет прием с 8.00 до 10.30 по понедельникам и четвергам. Напечатать список дней заданного месяца года, когда этот врач ведет прием в первую смену (учитывать только рабочие дни).

9. В учебном отделе научно-технической библиотеки ДонГТУ последний вторник каждого месяца - санитарный день. Получить даты всех месяцев заданного года, на которые приходится санитарный день.

10. Весенний семестр в ДонГТУ начинается 10 февраля. На первом курсе для специальности "Информационные системы в менеджменте" этот семестр длится 16 недель. Определить, в какой день недели закончится весенний семестр в заданном году.

11. Определить, на какой день недели приходится 8 Марта в ближайшем высокосном году, следующем за заданным.

12. Получить даты всех праздничных дней года, которые совпали в заданном году с выходными (субботой или воскресеньем).

13. Вычислить количество пятниц, приходящихся на 13-е числа, в ближайшие три года, начиная от заданного.

14. Журналист ведет телевизионную передачу по вторникам и пятницам. Составить график выхода в эфир этого журналиста на два месяца, предшествующие заданному.

15. Рекламный еженедельник выходит в свет по четвергам. Определить номер последнего вышедшего печатного издания, если известна сегодняшняя дата.

16. Подсчитать, сколько прошло воскресений от начала текущего квартала до последнего праздничного дня этого квартала. Известна текущая дата d.m.g.

17. Учебная практика для студентов специальности "Информационные системы в менеджменте" в ДонГТУ начинается k-го июня и длится 3 недели. Вступительные экзамены в ДонГТУ начинаются 1 июля. Сколько дней учебной практики в заданном году совпадают со вступительными экзаменами (учитывать только рабочие дни)?

18. День Радио отмечается ежегодно 7 мая. Определить, на какие дни недели приходится этот день в годы, считая от начала текущего десятилетия до заданного года.

19. День кафедры "Прикладной математики и информатики" проводится во вторую пятницу апреля. Определить даты последних трех Дней кафедры, если известна текущая дата d.m.g.

20. Задана дата d.m.g. Сколько полных кварталов осталось до конца этого года?

21. Телепрограмма выходит в эфир по средам в рабочие дни. Сколько выпусков этой программы будет в заданном году?

22. В учебном отделе научно-технической библиотеки ДонГТУ последняя среда каждого месяца - санитарный день. Получить даты всех санитарных дней этого отдела от заданной даты до конца этого года.

23. Н.А. Некрасов родился 10 декабря 1821г. Когда в последний раз день рождения Н.А. Некрасова приходился на субботу, если известна текущая дата d.m.g?

24. День авиации и космонавтики отмечается ежегодно 12 апреля. Приходится ли в заданном году этот праздник на день недели?

25. Сколько раз в заданном году начало месяца приходится на вторник?

26. Врач ведет прием с 10.30 до 13.00 по средам и пятницам. Сформировать список дней заданного месяца года, когда этот врач ведет прием в рабочие дни во вторую смену.

27. Определить, на какой день недели приходится День Победы (9 Мая) в ближайшем высокосном году, предшествующем заданному.

28. Осенний семестр в ДонГТУ начинается 1 сентября и длится для студентов специальности "Информационные системы в менеджменте" 18 недель. Напечатать список всех праздничных дней, которые приходятся на этот семестр и соответствующие им дни недели (для заданного года).

29. На какие дни недели приходятся в заданном году даты начала кварталов?

30. Диктор ведет радиопередачу по понедельникам, средам и пятницам в рабочие дни. Составить график выхода в эфир этого диктора на ближайшие три недели, начиная от заданной даты.

31. Журнал "Огонёк" выходит в свет по вторникам. Определить номера этого еженедельника, начиная от заданной даты до конца текущего месяца.

32. Сессия начинается в день d.m.g.. Для студентов специальности "Информационные системы в менеджменте" она состоит из N экзаменов. Первый экзамен приходится на второй день сессии. Между экзаменами - M дней подготовки. Сформировать расписание экзаменов сессии в заданном году.

33. Зимняя сессия начинается в день d.m.g.. Консультация перед экзаменом проводится в день, предшествующий экзамену, если экзамен не приходится на понедельник. В противном случае консультация проводится в субботу, предшествующую экзамену в понедельник. Для студентов специальности "Информационные системы в менеджменте" сессия состоит из N экзаменов. Первый экзамен приходится на третий день сессии, между экзаменами - M дней подготовки. Сформировать расписание консультаций сессии в заданном году.

34. Первый вступительный экзамен в ДонГТУ проводится 1 июля. Далее экзамены проводятся с интервалом N дней. Определить дни недели, на которые приходятся вступительные экзамены в заданном году (всего экзаменов M).

35. Каких рабочих дней в текущем месяце больше - тех, которые уже прошли, или тех, которые еще будут, если задана текущая дата d.m.g?

36. В Украине понедельник, следующий за праздничным днем, который приходится на выходной (субботу или воскресенье), считается нерабочим днем. Определить, есть ли в заданном месяце такие нерабочие понедельники.

37. Лекции в весеннем семестре по дисциплине "Технология программирования" проводятся по вторникам еженедельно и по пятницам верхней недели. Определить порядковые номера лекций

Методические указания и задания к лабораторным работам по курсу "Основы программирования и алгоритмические языки" (для студентов специальности "Программное обеспечение автоматизированных систем") и "Технология программирования" (для студентов специальности "Информационные системы в менеджменте"). Ч.1/ Сост. Н.Н. Дацун, И.А. Коломойцева. - Донецк, ДонГТУ, 1997. - 132 с.

Даны задания и указания к их выполнению по следующим темам: календарные задачи; решение задач аналитической геометрии; работа с многочленами и матрицами; работа с символьными и текстовыми данными; работа с табличными данными. Для каждой темы приведен пример и этапы его решения на ЭВМ: постановка задачи; метод решения; алгоритм; программа на языке Си.

В подготовке материалов методических указаний принимали участие студенты групп МИ9баб, ПО9баб.

Составители	Н.Н. Дацун, доц. И.А. Коломойцева, асс.
Отв. за выпуск	Е.А. Башков, проф.
Рецензент	В.Н. Павлыш, доц.

1 ОСНОВНЫЕ ПОЛОЖЕНИЯ

Цель дисциплины: дать специалисту необходимые знания о технологии программирования на алгоритмических языках и решении задач с помощью персональных электронно-вычислительных машин (ПЭВМ).

В результате изучения дисциплины студент должен знать основные технологические этапы решения задач на ЭВМ, средства подготовки, алгоритмизации и структурного программирования задач на языках программирования Си и Паскаль.

Студент должен уметь:

- пользоваться аппаратным обеспечением ПЭВМ с разнообразным периферийным оборудованием (клавиатурой, дисплеем, накопителями и др.) на уровне квалификации "оператор ЭВМ";
- использовать средства операционных систем и программ сервисного обеспечения для автоматизации работы с ПЭВМ;
- выполнять технологические операции по автоматизированной подготовке, отладке и выполнению программ на ПЭВМ и обрабатывать результаты вычислений.

1.1 Содержание практической подготовки и формы контроля

Дисциплина "Технология программирования" имеет в своем составе лабораторные работы как вид учебной работы. Лабораторные работы являются способом закрепления знаний, полученных студентами на лекциях и во время самостоятельной подготовки, а также способом привития навыков выполнения инженерных и научных расчетов с использованием ПЭВМ. Эти виды занятий проводятся в специализированных лабораториях программирования на установленных там ПЭВМ.

Как формы контроля качества полученных знаний используются:

- контроль с помощью обучающих программ непосредственно на ПЭВМ;

- опрос во время защиты отчетов по лабораторным работам непосредственно на ПЭВМ;
- опрос во время защиты отчета по курсовой работе непосредственно на ПЭВМ;
- опросы в течение семестра.

Изучение дисциплины завершается сдачей экзамена непосредственно на ПЭВМ.

Занятия при проведении лабораторных работ организованы таким образом, что студент по методическим указаниям к работам, конспекту лекций и рекомендованной литературе в течение семестра самостоятельно готовится к аудиторным занятиям, а на занятиях выполняет индивидуальные задания под руководством преподавателя. Распределение тем лабораторных работ по семестрам приведено в таблице 1.1:

Таблица 1.1 - Темы лабораторных работ

N ра- бо- ты	N се- мес- тра	Тема
1	1	Календарные задачи на языке Си
2	1	Решение задач аналитической геометрии на языке Си
3	1	Работа с многочленами и матрицами на языке Си
4	1	Работа с символьными и текстовыми данными на языке Си
5	1	Работа с табличными данными на языке Си
6	2	Рекурсия в языке Си
7	2	Работа с файлами последовательного доступа на языке Си
8	2	Графические средства Turbo-C
9	2	Работа с динамическими структурами данных на языке Си
10	2	Работа с файлами прямого доступа на языке Си
11	2	Календарные задачи на языке Паскаль
12	2	Решение задач аналитической геометрии на языке Паскаль
13	2	Работа с многочленами и матрицами на языке Паскаль

1.2 Требования к оформлению отчета по лабораторным работам

1. Материалы по выполненным лабораторным работам студент оформляет в виде отчетов. Содержание отчета оговаривается в каждой главе, соответствующей теме лабораторной работы.

2. Отчет должен быть предоставлен в рукописном или машинописном (в виде распечатки на принтере) виде на листах формата А4.

3. Текст отчета должен быть подготовлен в соответствии с Государственным стандартом Украины ДСТУ 3008-95 "Документация. Отчеты в сфере науки и техники. Структура и правила оформления".

4. Отдельные части отчета, соответствующие этапам решения задач на ЭВМ, должны быть представлены студентом на проверку согласно графика выполнения лабораторных работ.

5. Все части отчета должны быть снабжены пояснениями, отражающими семантику переменных, фрагментов алгоритма и т.д.

6. Таблица трассировки должна содержать столбцы для всех:

- исходных данных;
- промежуточных данных;
- условий, вычисляемых в программе;
- результатов.

Количество тестов должно быть достаточным для проверки всех ограничений и возможных вариантов результатов.

7. Текст программы должен содержать обязательные комментарии:

- название отчета (номер, тема лабораторной работы, номер варианта);
- фамилия и инициалы студента, название группы;
- условие решаемой задачи;
- название разделов программы: "декларации" (с указанием входных, выходных, промежуточных переменных), "ввод данных", "проверка ограничений", "исполнимая часть", "вывод результатов";

- описание семантики:

- всех функций, определенных пользователем,
- всех переменных,
- шагов алгоритма,
- вычисляемых признаков и значений промежуточных переменных.

8. Текст программы должен быть структурирован с обязательными отступами для вложенных конструкций.

9. При использовании вспомогательных алгоритмов текст функций, определенных пользователем, не должен превышать размера экрана (до 25 строк).

10. Программный документ "Описание программы" должен быть выполнен в соответствии с ГОСТом ЕСПД.

11. Отчет должен обязательно содержать выходы, которые характеризуют выбранный метод решения задачи, использованное программное обеспечение, особенности и результаты решения задачи на ПЭВМ.

12. Отчет студент защищает, как правило, перед выполнением следующей работы.

1.3. Требования к интерфейсу программы

1.3.1. Организация входных данных

1. Все исходные данные должны быть введены с системного устройства ввода (клавиатура).

В том случае, если тестовые примеры предполагают большой объем работы по вводу данных (массивов, матриц, текстов, таблиц), необходимо подготовить соответствующее количество тестовых файлов данных. При отладке, тестировании и демонстрации лабораторной работы на ПЭВМ студент должен переадресовать системный ввод с клавиатуры на необходимый файл.

Ввод исходных данных должен сопровождаться обязательными приглашениями к вводу, которые поясняют вводимую информацию; например, при вводе массива или матрицы указывают имя объекта и индексы вводимого элемента, при вводе структуры — наименование ее членов.

1.3.2 Организация выходных данных.

1. Все данные должны быть выведены на системное устройство вывода (экран монитора). Если общий объем выводимых данных превышает размер экрана в текстовом режиме (25 строк по 80 символов), то при отладке, тестировании и демонстрации лабораторной работы на ПЭВМ студент должен переадресовать системный вывод с экрана в файл во внешней памяти.

2. Программа в начале своей работы должна очистить экран и вывести в первых строках экрана следующую информацию:

- номер и тема лабораторной работы;
- фамилия и инициалы студента, название группы;
- условие решаемой задачи.

3. Затем, должны быть обязательно выведены в сопровождении семантического комментария исходные данные, введенные пользователем.

4. Исходные данные и результаты должны быть выведены в удобочитаемом виде в соответствии со своей семантикой: матрицы по строкам, табличные данные в виде таблицы построчно с наименованием ее столбцов и т.п.

5. Результаты работы отдельных частей алгоритма необходимо выводить на экран с задержкой изображения для просмотра, используя для продолжения работы ввод ответа пользователя.

1.4 Перечень используемого программного обеспечения

Перечень используемого программного обеспечения приведен в таблице 1.2.

1.5 Состав используемых технических средств лабораторной базы

Во время изучения дисциплины для выполнения лабораторных работ используются лаборатории вычислительной техники кафедры ПМиИ на базе ПЭВМ, в состав которых входят IBM PC/AT, объединенные в локальную сеть кафедры, соединенную с сетью университета.

по этой дисциплине, которые состоятся в заданном месяце. Весенний семестр начинается 10 февраля и длится 16 недель. Первая неделя семестра считается верхней.

38. Есть ли в текущем квартале (для заданной даты года) месяцы, которые заканчиваются в воскресенье, и сколько их?

39. Определить дни недели для первых чисел всех месяцев заданного года.

40. Газета выходит в свет по средам и пятницам. Определить дату выхода заданного номера этого издания в заданном году.

2.4 Контрольные вопросы по лабораторной работе

- 1) Реализовать алгоритм "вечного календаря".
- 2) Организовать на языке Си ввод списка праздничных дат текущего года.
- 3) Организовать на языке Си вывод списка рабочих дней заданного месяца.

2.5 Содержание отчета

1. Условие задачи.
2. Математические расчеты и вывод формул для раздела "Связь" постановки задачи.
3. Постановка задачи.
4. Метод решения задачи.
5. Алгоритм.
6. Текст программы на языке Си.
7. Таблица трассировки программы.
8. Графическая интерпретация тестовых примеров (с изображением формируемых массивов, их указателей и т.д.).
9. Выводы.
10. Программный документ "Описание программы".

3 ЛАБОРАТОРНАЯ РАБОТА N 2. РЕШЕНИЕ ЗАДАЧ АНАЛИТИЧЕСКОЙ ГЕОМЕТРИИ НА ЯЗЫКЕ СИ

Цель работы: освоить приемы алгоритмизации и программирования при работе с данными вещественного типа и со стандартной библиотекой математических функций; с приемами моделирования геометрических объектов на плоскости структурами данных типа "массив" (на примере решения задач аналитической геометрии).

Контрольные вопросы:

- 1) Как настраивается интегрированная среда Turbo-C на работу с плавающей арифметикой?
- 2) Какая стандартная библиотека поддерживает работу с математическими функциями?
- 3) Какой тип данных у аргументов и результатов математических функций этой стандартной библиотеки?

3.1 Приемы алгоритмизации и программирования при решении задач аналитической геометрии

При выборе структуры данных для представления в алгоритмическом языке геометрических объектов следует учитывать следующее.

3.1.1 Представление отдельного геометрического объекта

1) Отдельный геометрический объект может быть представлен массивом значений его параметров:

- "точка" -> массив из двух элементов, первый из которых соответствует координате по оси абсцисс, второй - координате по оси ординат;

- "окружность" -> массив из трех элементов, первые два из которых соответствуют центру окружности (см. объект "точка"), а третий - величине радиуса;

- "прямая"

а) в случае представления прямой с помощью уравнения $y=kx+b$ используется массив из двух элементов, которые соответствуют значениям k и b ;

б) в случае представления с помощью уравнения прямой, проходящей через две точки, используется массив из четырех элементов, которые попарно соответствуют этим точкам (см. объект "точка");

- "гипербола", "эллипс" -> массив из двух элементов, которые соответствуют коэффициентам канонического уравнения соответствующего геометрического объекта.

3.1.2 Представление множества геометрических объектов

Множество геометрических объектов может быть представлено массивом, количество элементов которого по первому измерению соответствует количеству объектов множества.

Так для представления множества точек достаточно использовать массив

точки: массив [1..N, 1..2] вещественных,
где N-количество точек в множестве.

Тогда значение элемента *точки[i, 1]* соответствует значению координаты *x* i-й точки множества, а значение элемента *точки[i, 2]* - значению координаты *y* этой точки.

3.1.3 Особенности представления геометрических объектов в языке Си.

При моделировании геометрических объектов средствами языка Си необходимо учесть следующее:

1) Выделение памяти для массива, моделирующего множество геометрических объектов, в программе на языке Си выполняют статически с использованием именованных констант:

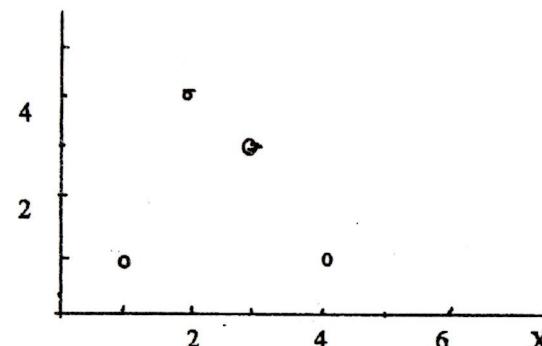
```
#define N 30
main()
float points [N][2]; {множество точек}
```

2) Реальное количество элементов множества геометрических объектов запрашивают у пользователя, а затем проверяют введенное значение не только на ограничения, определенные условием задачи, но и на ограничения, связанные с выделением памяти:

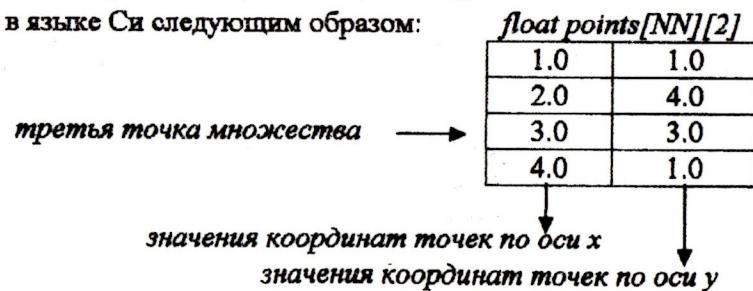
```
#define NN 30
main()
{
    float points[NN][2];
    int n; /* реальное количество элементов*/
    printf ("Введите количество элементов множества");
    printf ("(< %d) :\n", NN);
    scanf ("%d", &n);
    /* проверка ограничений: */
    if ((n < 2) /* ограничения условия задачи */
        || (n > NN)) /* ограничения выделения памяти */
        printf ("Вы ошиблись при вводе!!!\n");
}
```

3) Доступ к элементам множества геометрических объектов выполняется индексированием соответствующего массива по первому измерению, которое соответствует порядковому номеру объекта в множестве.

Например, множество точек А: {(1,1), (2,4), (3,3), (4,1)} моделируем в языке Си следующим образом:



лируем в языке Си следующим образом:



Доступ к i-й точке:

points[i-1][0] - координата точки по оси x;

points[i-1][1] - координата точки по оси y.

Следует помнить особенность индексации массивов в языке Си: младший индекс по любому измерению равен нулю. Это означает, что в программе на языке Си при доступе к элементу структуры данных типа "массив" индекс по каждому из измерений должен быть младше на единицу по сравнению с языком записи метода или языком записи алгоритмов.

3.1.4 Особенности работы с данными вещественного типа в языке Си

3.1.4.1 Настройка среды Turbo-C 2.0 на работу с плавающей арифметикой

Для работы с плавающей арифметикой необходимо произвести настройку среды Turbo-C 2.0. Для этого в главном меню среды выбирают опцию "Options" и нажимают клавишу *ENTER*. В появившемся вертикальном меню первого уровня выбирают опцию "Compiler" и снова нажимают клавишу *ENTER*. В следующем появившемся вертикальном меню второго уровня выбирают опцию "Code generation" и снова нажимают клавишу *ENTER*. В вертикальном меню третьего уровня выбирают опцию "Floating point". Нажатием клавиши *ENTER* "прокручивают" возможные значения реализации на данном компьютере плавающей арифметики и выбирают подходящую опцию:

Таблица 3.1 - Варианты настройки Turbo-C на работу с плавающей арифметикой

Значения опции "Floating point"	Вариант настройки Среды Turbo-C
None	Отключить плавающую арифметику
Emulation	Моделировать работу с плавающей арифметикой при отсутствии в составе компьютера сопроцессора
8087/80287	Выполнять работу с плавающей арифметикой с помощью сопроцессора

Выбранное значение настройки должно остаться в окне значения опции "Floating point".

Затем, дважды нажимая клавишу *ESC*, закрывают последовательно вертикальные меню третьего и второго уровня. В вертикальном меню первого уровня выбирают опцию "Save options" и нажимают клавишу *ENTER*. После открытия окна "Config File" нажимают клавишу *ENTER*. После появления окна "Verify" с текстом "Overwrite TCCONFIG.TC (Y/N)" подтвердите выбор (т.е. сохранение настройки на работу с плавающей арифметикой) нажатием клавиши *Y*.

После выполнения сохранения выполненной настройки вертикальное меню первого уровня автоматически закрывается, и курсор возвращается в главное меню среды Turbo-C.

3.1.4.2 Ввод значений элементов вещественного массива в Turbo-C 2.0

При вводе в цикле значений элементов массива вещественного типа в Turbo-C 2.0 следует поступить следующим образом:

1) объявить скалярную рабочую переменную вещественного типа;

2) осуществить в цикле ввод значения в эту переменную с последующим присваиванием этого значения соответствующему элементу массива:

```
#define NN 30
main()
{
    float points[NN][2]; /* множество точек*/
    int n, i; /*количество точек и порядковый номер точки*/
    float r; /*вспомогательная переменная */
    ...
    for (i=0; i<n; i++)
    {
        printf ("Введите координату x %d-й точки:", i+1);
        scanf ("%f", &r);
        points[i][0] = r;
        printf ("Введите координату y %d-й точки:", i+1);
        scanf ("%f", &r);
        points[i][1] = r;
    };
}
```

3.1.5 Технология организации ввода большого количества данных в языке Си

Если при тестировании (демонстрации) программы необходимо вводить с системного устройства ввода (клавиатуры) большое количество данных, то следует изменить процесс ввода этой информации.

1) Заранее следует подготовить достаточное количество файлов, с тестами в соответствии с действиями по вводу данных.

2) При вызове исполнямого файла выполнить переназначение системного устройства ввода на этот файл.

Для примера с множеством А из четырех точек (см. 3.1.3, пп.3), данные для которого вводятся по программе из 3.1.4.2, такой файл имеет вид:

```
1 1 2 4 3 3 4 1
```

файл *test1*

Пусть имя исполняемого файла этой программы *lab2.exe*. Тогда вызов этой программы выглядит так:

```
lab2.exe < test1
```

После этого данные для программы *lab2* будут вводиться из файла *test1*.

3.1.6 Технология организации вывода большого количества данных в языке Си

Если при тестировании (демонстрации) программы необходимо выводить на системное устройства вывода (экран) большое количество данных, то при вызове исполнямого файла необходимо выполнить переназначение системного устройства вывода в некоторый файл.

Пусть имя исполняемого файла программы *lab2.exe*. Тогда вызов этой программы выглядит так:

```
lab2.exe > res
```

После этого выходные данные программы *lab2* будут выводиться в файл *res*.

3.1.7 Организация вывода сообщений и результатов в языке Си при использовании функций форматированного вывода

В языке Си несколько строковых констант, записанных рядом, конкатенируются. Этим свойством языка следует пользоваться при записи строки формата, если длина последней превышает размер строки экрана. Например, вывод на экран геометрического объекта "прямая, проходящая через две точки" может быть организован следующим образом:

```
float pr[4]; ...  
printf ("Прямая, проходящая через две точки"  
" (%f, %f) и (%f, %f):",  
pr[0], pr[1], pr[2], pr[3]);
```

3.1.8 Некоторые вопросы эффективности программ

Для повышения эффективности программы и читаемости ее текста не следует повторять многократные вычисления значений переменных с индексом, соответствующих параметрам геометрических объектов. Придерживайтесь следующих правил.

1) Параметры геометрических объектов как исходные данные:

- один раз вычислить значения переменных с индексами и присвоить их промежуточным скалярным переменным;
- в дальнейшем в программе использовать именно эти скалярные переменные при анализе условий и вычислении результатов.

2) Параметры геометрических объектов как выходные данные:

- сформировать значения результатов в промежуточных скалярных переменных;
- один раз присвоить значения этих скалярных переменных соответствующим переменным с индексами (см. решение примера из пп.3.3).

3.2 Использование в языке Си математических функций

Работу с математическими функциями в языке Си обеспечивает стандартная библиотека *math.h*. Все функции этой библиотеки имеют тип аргумента (-ов) и тип результата *double*.

Ниже приведены описания функций этой библиотеки.

3.2.1 Тригонометрические, обратные тригонометрические и гиперболические функции

3.2.1.1 Описание семантики функций

```
double acos (double x);           /* вычисляет арккосинус x*/
double asin (double x);           /* вычисляет арксинус x*/
double atan (double x);           /* вычисляет арктангенс x*/
double atan2 (double y, double x); /* вычисляет арктангенс y/x*/
double cos (double x);            /* вычисляет косинус x*/
double cosh (double x);           /* вычисляет косинус гиперболический x*/
double sin (double x);            /* вычисляет синус x*/
double sinh (double x);           /* вычисляет синус гиперболический x*/
double tan (double x);            /* вычисляет тангенс x*/
double tanh (double x);           /* вычисляет тангенс гиперболический x*/
```

3.2.1.2 Ошибки этапа выполнения для этих функций

TLOSS /* аргумент слишком велик, например, sin(10e70) */
(см. также ошибкипп.3.2.2.2: DOMAIN, SING, OVERFLOW,
UNDERFLOW).

3.2.2 Трансцендентные функции

3.2.2.1 Описание семантики функций

```
double exp (double x); /* экспонента  $e^x$ ; возвращает 0.0 при
потере значимости */
double ldexp (double x, int exponent); /* вычисляет длинное
вещественное значение результата возведения в степень, т.е.
значение функции  $x \cdot 2^{exponent}$ */
double log (double x); /* возвращает натуральный логарифм x*/
double log10 (double x); /* возвращает десятичный логарифм x*/
double pow (double x, double y); /* вычисляет значение  $x^y$ */
double sqrt (double x); /* вычисляет значение  $\sqrt{x}$ */
```

3.2.2.2 Ошибки этапа выполнения для этих функций

DOMAIN /* аргумент log отрицательный */
SING /* аргумент некорректен, например, pow(0, -2) */
OVERFLOW /* переполнение порядка, например, exp(1000) */
UNDERFLOW /* исчезновение порядка, например exp(-1000) */

3.2.3 Функции вычисления абсолютного значения

Все приведенные ниже функции вычисляют абсолютное значение своего аргумента. Значение типа результата этих функций совпадает с типом аргумента.

```
int abs (int x);
long labs (long x);
double fabs (double x);
```

3.2.4 Прочие функции

double ceil (double x); /* округлить: возвращает наименьшее
целое, не меньшее чем x (округляет x до следующего большего
целого числа) */

double floor (double x); /* округлить: возвращает наибольшее
целое, не превышающее значения x (округляет x до ближайшего
меньшего значения) */

double fmod (double x, double y); /* возвращает значение, равное
остатку от деления x на y*/

double frexp (double x, int *exponent); /* разбить число с
плавающей точкой на мантиссу и показатель степени:
возвращает мантиссу m значения с плавающей точкой, такую
что $0.5 \leq m < 1$; запоминает показатель степени n типа int в
переменную, адресуемую указателем exponent */

double modf (double x, double *ipart); /* разбить число с
плавающей точкой на целую и дробную части: возвращает
дробную часть x; запоминает значение целой части по адресу,
содержащемуся в ipart */

3.3 Использование в языке Си математических констант

Стандартная библиотека *math.h* языка Си обеспечивает работу
с математическими константами. Используйте именованные
константы этой библиотеки для обозначения математических
констант.

Ниже приведены описания некоторых из этих констант (точность всех макросов - 21 знак).

Таблица 3.1 - Макросы для математических констант из math.h

Именованная константа в Си	Значение	Математический смысл
M_E	2.71828182845904523536	e
M_LOG2E	1.44269504088896340736	$\log_2 e$
M_LOG10E	0.434294481903251827651	$\log_{10} e$
M_LN2	0.693147180559945309417	$\ln 2$
M_LN10	2.30258509299404568402	$\ln 10$
M_PI	3.14159265358979323846	π
M_PI_2	1.57079632679489661923	$\pi / 2$
M_PI_4	0.785398163397448309116	$\pi / 4$
M_1_SQRTPI	0.564189583547756286948	$\sqrt{\pi}$
M_SQRT2	1.41421356237309504880	$\sqrt{2}$
M_SQRT_2	0.707106781186547524401	$\sqrt{2} / 2$

3.4 Пример

Построить треугольник, вершинами которого являются середины сторон исходного треугольника ABC, и определить его площадь.

3.4.1 Постановка задачи

3.4.1.1 Исходные данные

tr: массив [1..3, 1..2] вещественных; { треугольник }

3.4.1.2 Ограничения

{условие невырожденности треугольника}

$$\begin{aligned} |tr_{1,j} - tr_{2,j}| + |tr_{2,j} - tr_{3,j}| &< |tr_{1,j} - tr_{3,j}| \wedge \\ |tr_{2,j} - tr_{3,j}| + |tr_{3,j} - tr_{1,j}| &< |tr_{1,j} - tr_{2,j}| \wedge \\ |tr_{3,j} - tr_{1,j}| + |tr_{1,j} - tr_{2,j}| &< |tr_{2,j} - tr_{3,j}|; \quad j \in [1, 2] \end{aligned}$$

3.4.1.3 Результаты

сообщ: строка[1..60]; { сообщение об ошибках}

res: массив [1..3, 1..2] вещественных; { треугольник }

s: вещественное; { площадь }

3.4.1.4 Связь

Координаты точек треугольника-результата формируем как среднее координат соответствующих сторон:

$$\begin{aligned} x_{a1} &= \frac{x_a + x_b}{2}; \\ y_{a1} &= \frac{y_a + y_b}{2}; \\ x_{b1} &= \frac{x_b + x_c}{2}; \\ y_{b1} &= \frac{y_b + y_c}{2}; \\ x_{c1} &= \frac{x_a + x_c}{2}; \\ y_{c1} &= \frac{y_a + y_c}{2}; \end{aligned}$$

Площадь треугольника вычисляется по формуле:

$$s = \frac{1}{2} abs \begin{vmatrix} x_{b1} - x_{a1} & x_{c1} - x_{a1} \\ y_{b1} - y_{a1} & y_{c1} - y_{a1} \end{vmatrix}$$

3.4.2. Метод решения

$$s := \frac{1}{2} * |((x_{b1}-x_{a1})*(y_{c1}-y_{a1}) - (x_{c1}-x_{a1})*(y_{b1}-y_{a1}))|$$

$$res_{1,1} := x_{a1} \qquad \qquad res_{1,2} := y_{a1}$$

$$res_{2,1} := x_{b1} \qquad \qquad res_{2,2} := y_{b1}$$

$$res_{3,1} := x_{c1} \qquad \qquad res_{3,2} := y_{c1}$$

сообщ := "треугольник
вырожден"

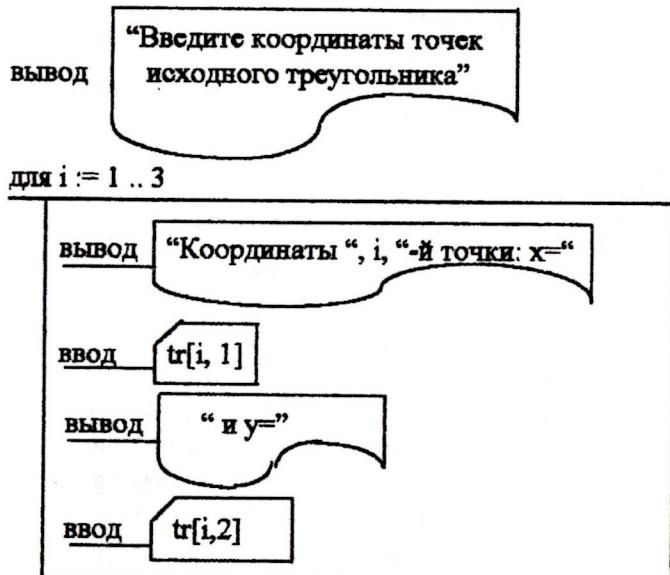
$$\begin{aligned} \text{не } (& |xa-xb| + |xb-xc| < |xa-xc| \text{ и} \\ & |xb-xc| + |xo-xa| < |xa-xb| \text{ и} \\ & |xc-xa| + |xa-xb| < |xb-xc| \text{ и} \\ & |ya-yb| + |yb-yc| < |ya-yc| \text{ и} \\ & |yb-yc| + |yc-ya| < |ya-yb| \text{ и} \\ & |yc-ya| + |ya-yb| < |yb-yc| \text{ и} \end{aligned}$$

$$\begin{aligned}
 xc1 &:= \frac{1}{2} (xa + xc) & yc1 &:= \frac{1}{2} (ya + yc) \\
 xb1 &:= \frac{1}{2} (xb + xc) & yb1 &:= \frac{1}{2} (yb + yc) \\
 xa1 &:= \frac{1}{2} (xa + xb) & ya1 &:= \frac{1}{2} (ya + yb) \\
 xa &:= tr[1,1] & ya &:= tr[1,2] \\
 xb &:= tr[2,1] & yb &:= tr[2,2] \\
 xc &:= tr[3,1] & yc &:= tr[3,2]
 \end{aligned}$$

Промежуточные данные:

xa, ya, {координаты точки A исходного треугольника}
 xb, yb, {координаты точки B исходного треугольника}
 xc, yc: вещественные; {координаты точки C исходного треугольника}
 xa1, ya1, {координаты точки A треугольника-результата }
 xb1, yb1, {координаты точки B треугольника-результата }
 xc1, yc1: вещественные; {координаты точки C треугольника-результата }

3.4.3 Алгоритм



{ вычисление значений скалярных переменных, соответствующих исходным данным }

xa := tr[1,1]

ya := tr[1,2]

xb := tr[2,1]

yb := tr[2,2]

xc := tr[3,1]

yc := tr[3,2]

{ проверка ограничений }

если abs(xa-xb) + abs(xb-xc) < abs(xc-xa) и
 abs(xb-xc) + abs(xc-xa) < abs(xa-xb) и
 abs(xc-xa) + abs(xa-xb) < abs(xb-xc) и
 abs(ya-xb) + abs(xb-xc) < abs(xc-xa) и
 abs(xb-xc) + abs(xc-xa) < abs(xa-xb) и
 abs(xc-xa) + abs(xa-xb) < abs(xb-xc) то

{ данные корректны }

xa1 := (xa+xb)/2

ya1 := (ya+yb)/2

xb1 := (xb+xc)/2

yb1 := (yb+yc)/2

xc1 := (xa+xc)/2

yc1 := (ya+yc)/2

s := abs((xb1-xa1)*(yc1-ya1) - (xc1-xa1)*(yb1-ya1))/2

res[1,1] := xa1

res[1,2] := ya1

res[2,1] := xb1

res[2,2] := yb1

res[3,1] := xc1

res[3,2] := yc1

вывод
“Площадь треугольника ((“, xa1, “, ya1, “), (“, xb1, “, yb1, “), (“, xc1, “, yc1, “)) равна „, s’”

иначе

вывод

“Исходный треугольник ((“, xa, “, “, ya, “), (“, xb, “, “, yb, “), (“, xc, “, “, yc “)) вырожден”

3.4.4 Текст программы на языке Си

```
/* ===== РЕШЕНИЕ ЗАДАЧ АНАЛИТИЧЕСКОЙ ГЕОМЕТРИИ ===== */
/*          ЛАБОРАТОРНАЯ РАБОТА N 2           */
/*          студентки гр. МИ93 Ивановой С.      */
/* Построить треугольник, вершинами которого являются */
/* середины сторон исходного треугольника АВС, и определить */
/* его площадь. */
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
main()
{
/* ===== ИСХОДНЫЕ ДАННЫЕ ===== */
float tr[3][2];           /* исходный треугольник */
/* ===== ВЫХОДНЫЕ ДАННЫЕ ===== */
float res[3][2];           /* треугольник-результат */
float s;                   /* площадь треугольника-результата */
/* ===== ПРОМЕЖУТОЧНЫЕ ДАННЫЕ ===== */
float xa, ya, /* координаты точки А исходного треугольника */
      xb, yb, /* координаты точки В исходного треугольника */
      xc, yc; /* координаты точки С исходного треугольника */
float xl, yl, /* координаты точки А треугольника-результата */
      xl, yl, /* координаты точки В треугольника-результата */
      xl, yl; /* координаты точки С треугольника-результата */
float r;                 /* рабочая переменная для ввода данных */
int i;                   /* порядковый номер точки исходного треугольника */
```

```
/* == ВВОД ДАННЫХ И ПРОВЕРКА ОГРАНИЧЕНИЙ == */
clrscr();
printf("Введите координаты точек исходного треугольника:");
for (i=0; i<3; i++)
{
    printf("\nкоординаты: %d-й точки: x=", i+1);
    scanf("%f", &r);
    tr[i][0]=r;                                /* координата по оси x */
    printf("y=");
    scanf("%f", &r);
    tr[i][1]=r;                                /* координата по оси y */
}
/* формирование промежуточных скалярных переменных */
xa=tr[0][0];           ya=tr[0][1];
xb=tr[1][0];           yb=tr[1][1];
xc=tr[2][0];           yc=tr[2][1];
/* ===== проверка ограничений ===== */
if (fabs(xa-xb)+fabs(xb-xc)<fabs(xc-xa) ||
    fabs(xb-xc)+fabs(xc-xa)<fabs(xa-xb) ||
    fabs(xc-xa)+fabs(xa-xb)<fabs(xb-xc) ||
    fabs(ya-yb)+fabs(yb-yc)<fabs(yc-ya) ||
    fabs(yb-yc)+fabs(yc-ya)<fabs(ya-yb) ||
    fabs(yc-ya)+fabs(ya-yb)<fabs(yb-yc))
)
printf("Исходный треугольник ((%f, %f), (%f, %f), "
       " (%f, %f)) вырожден\n", xa, ya, xb, yb, xc, yc );
else /* вычисление результатов */
{
    xl=(xa+xb)/2;
    yl=(ya+yb)/2;
    xl=(xb+xc)/2;
    yl=(yb+yc)/2;
    xl=(xa+xc)/2;
    yl=(ya+yc)/2;
    s=fabs ((xl-xa)*(yl-ya)-(xl-ya)*(yl-xa))/2;
    res[0][0]=xl;   res[0][1]=yl;
    res[1][0]=xl;   res[1][1]=yl;
    res[2][0]=xl;   res[2][1]=yl;
```

```

/* ===== вывод результатов ===== */
printf("Площадь треугольника("
    "(%f, %f), (%f, %f), (%f, %f)) равна %f\n",
    xa1, ya1, xb1, yb1, xc1, yc1, s);
getch();
}
}

```

3.5 Варианты заданий

1. Определить, образуют ли точки заданного множества квадрат со сторонами, параллельными осям координат.
2. Найти площадь произвольного четырехугольника (путем разбиения его на треугольники).
3. Найти точку пересечения медиан треугольника.
4. Определить, являются ли точки заданного множества вершинами ромба.
5. Найти площадь трапеции, основания которой параллельны осям координат.
6. Найти радиус окружности, вписанной в заданный равнобедренный треугольник.
7. Найти точку пересечения высот треугольника.
8. Найти площадь ромба.
9. Определить, являются ли точки заданного множества вершинами параллелограмма со сторонами, параллельными осям ОХ.
10. Найти площадь произвольного пятиугольника.
11. Найти радиус описанной окружности заданного треугольника.
12. Найти точку пересечения диагоналей параллелограмма.
13. Найти площадь правильного шестиугольника.
14. Определить, лежит ли точка внутри заданного круга.
15. Определить, принадлежат ли все три точки заданного множества одной прямой.
16. Четыре точки множества соединены попарно. Найти точку пересечения полученных таким образом прямых.
17. Определить, является ли заданный треугольник прямоугольным.

18. Определить, лежат ли все четыре точки заданного множества в первом квадранте.
19. Через две точки проведена прямая. Найти точку, также принадлежащую этой прямой и лежащую между исходными.
20. Определить, принадлежат ли все три точки заданного множества заданному эллипсу.
21. Определить, принадлежит ли точка заданной гиперболе.
22. Найти площадь той части круга, которая осталась после удаления из него заданного вписанного треугольника.
23. Определить, находится ли заданная окружность во втором квадранте.
24. Определить, лежит ли окружность $((X_0, Y_0), R_0)$ внутри окружности $((X_1, Y_1), R_1)$.
25. Найти площадь той части круга, которая осталась после удаления из него заданного вписанного круга.
26. Найти площадь той части треугольника ABC, которая осталась после удаления из него заданного треугольника $A_1B_1C_1$.
27. Определить, лежит ли точка внутри заданного круга.
28. Определить, какой квадрант плоскости содержит больше всего точек исходного множества.
29. Найти площадь круга, которая осталась после удаления из него заданного вписанного квадрата.
30. Есть ли в заданном множестве пересекающиеся окружности?
31. Определить, образуют ли 4 соединенные последовательно точки ромб.
32. В заданном множестве точек определить, какие точки образуют треугольник наименьшей площади.
33. Определить, есть ли в заданном множестве кругов круг с заданной площадью.
34. Сколько окружностей заданного множества являются концентрическими?
35. Задано множество прямых. Определить, в каком квадранте содержится наибольшее количество точек пересечения этих прямых.

36. Задано множество прямых. Сколько трапеций они образуют?
37. Определить, в каком квадранте находится заданная окружность.
38. Есть ли среди прямых, образованных попарным соединением точек заданного множества, прямые, параллельные осям координат?
39. Задано множество окружностей. Есть ли среди них такая, которая пересекается хотя бы с двумя другими окружностями этого же множества?
40. Задано множество окружностей и множество точек. Внутри какой окружности оказалось больше всего точек заданного множества?

3.6 Контрольные вопросы по лабораторной работе

- 1) Настроить среду Turbo-C на работу с плавающей арифметикой.
- 2) Записать на языке Си вычисление выражения

$$\ln 2 * \pi / 4 + e * \sqrt{2 * \log_{10} e}$$

с помощью математических констант стандартной библиотеки. Какой тип данных у результата?

- 3) Записать на языке Си вычисление выражения

$$\ln \operatorname{tg}(\bar{x}^{1/4} + e^{x/2})$$

с помощью математических функций стандартной библиотеки. Какой тип данных у аргументов и результата?

3.7 Содержание отчета.

1. Условие задачи.
2. Математические расчеты и вывод формул для раздела "Связь" постановки задачи.
3. Постановка задачи.
4. Метод решения задачи.
5. Алгоритм.
6. Текст программы на языке Си.
7. Таблица тестирования.
8. Геометрическая и графическая интерпретация тестовых примеров.
9. Выводы.
10. Программный документ "Описание программы".

4 ЛАБОРАТОРНАЯ РАБОТА N 3. РАБОТА С МНОГОЧЛЕНАМИ И МАТРИЦАМИ НА ЯЗЫКЕ СИ

Цель работы: освоить приемы алгоритмизации при организации разветвляющихся и повторяющихся вычислений (на примере решения задач с матрицами и многочленами) при использовании арифметики с плавающей запятой; приемы моделирования математических объектов "матрица" и "многочлен" структурами данных типа "массив".

Контрольные вопросы:

- 1) Какая структура данных языков программирования моделирует математический объект "матрица"?
- 2) Как организовано в современных алголоподобных языках программирования распределение памяти для многомерных массивов?
- 3) Существуют ли в языке Си операции и функции для работы со структурированным объектом "массив"?
- 4) Как выполняется инициализация многомерного массива в языке Си?

4.1 Приемы алгоритмизации и программирования при решении задач с многочленами

4.1.1 Представление математического объекта "многочлен" и организация работы с многочленами

Рассмотрим приемы алгоритмизации и программирования задач с многочленами на примере решения следующей задачи:

Найти сумму двух многочленов.

Многочлен представляется своими коэффициентами:

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \dots + a_n x^n, \quad (4.1)$$

т. о. многочлен степени n может быть промоделирован в языках программирования массивом коэффициентов a_0, a_1, \dots, a_n .

При сложении многочленов различных степеней необходимо просуммировать поэлементно их общую часть, а затем приписать коэффициенты, оставшиеся в многочлене с большей степенью. Здесь возможны три случая.

1) $n > m$.

Например:

$$P_n(x) = 25 + 2x - 1.5x^2 - x^3 + x^4; \quad n=4;$$

$$Q_m(x) = 1 - 3.7x + x^2; \quad m=2;$$

Находим степень многочлена-результата: $S = \max(n, m)$.

В нашем случае $S = n = 4$.

Находим степень общей части многочленов: $K = \min(n, m)$.

В нашем случае $K = m = 2$.

Представление исходных многочленов (степени n и m соответственно) и многочлена-результата:

степени: 0 1 2 3 4

$$P_n(x): \begin{array}{|c|c|c|c|c|} \hline 25 & 2 & -1.5 & -1 & 1 \\ \hline \end{array}$$

$$+ \quad Q_m(x): \begin{array}{|c|c|c|} \hline 1 & -3.7 & 1 \\ \hline \end{array}$$

$$R_s(x): \begin{array}{|c|c|c|c|c|} \hline 26 & -1.7 & -0.5 & -1 & 1 \\ \hline \end{array}$$

$\underbrace{\hspace{10em}}_{0 \dots K}$

$\underbrace{\hspace{10em}}_{1 \dots S}$

В этом случае $n > m$. Поэтому поэлементно суммируем $m+1$ коэффициент (при степенях от 0 до m) исходных многочленов $P_n(x)$ и $Q_m(x)$, а затем дописываем коэффициенты при степенях от $m+1$ до n из многочлена $P_n(x)$.

2) $n < m$.

Например:

$$P_n(x) = -9 + x - 3x^2; \quad n=2;$$

$$Q_m(x) = 90 + 2.5x + 7x^2 + 13x^3; \quad m=3;$$

Находим:

$$S = \max(n, m) = m = 3;$$

$$K = \min(n, m) = n = 2.$$

степени: 0 1 2

$$P_n(x): \begin{array}{|c|c|c|} \hline -9 & 1 & -3 \\ \hline \end{array}$$

$$+ \quad Q_m(x): \begin{array}{|c|c|c|c|} \hline 90 & 2.5 & 7 & 13 \\ \hline \end{array}$$

$$R_s(x): \begin{array}{|c|c|c|c|c|} \hline 81 & 3.5 & 4 & 13 \\ \hline \end{array}$$

$$\underbrace{\hspace{10em}}_{0 \dots K}$$

$$\underbrace{\hspace{10em}}_{0 \dots 1 \dots S}$$

В этом случае $n < m$. Поэтому поэлементно суммируем $n+1$ коэффициент (при степенях от 0 до n) исходных многочленов $P_n(x)$ и $Q_m(x)$, а затем дописываем коэффициенты при степенях от $n+1$ до m из многочлена $Q_m(x)$.

3) $n=m$.

Поэлементно суммируем n коэффициентов (при степенях от 0 до n) исходных многочленов P и Q .

Приведем этапы решения этой задачи на ЭВМ.

4.1.2 Постановка задачи

4.1.2.1 Исходные данные

п: целое; {степень $P_n(x)$ }

Р: массив [0..n] вещественных; {массив коэффициентов $P_n(x)$ }

m: целое; {степень $Q_m(x)$ }

Q: массив [0..m] вещественных; {массив коэффициентов $Q_m(x)$ }

4.1.2.2 Ограничения

$n \geq 0$

$m \geq 0$

4.1.2.3 Результаты:

S: целое ; {степень многочлена-результата}

R: массив [0..S] вещественных; {массив коэффициентов $R_i(x)$ }

4.1.2.4 Связь

Определяем наибольшую (S) и наименьшую (K) из степеней исходных многочленов. Суммируем попарно коэффициенты многочленов (до степени K), а затем дописываем оставшиеся S-K коэффициентов из многочлена с большей степенью.

4.1.3 Метод

$$R_i = Q_i \quad | \quad i = \overline{k+1, S} \quad | \quad m > n$$

$$R_i = P_i \quad | \quad i = \overline{k+1, S} \quad | \quad n > m$$

$$R_i = P_i + Q_i \quad | \quad i = \overline{0, k}$$

$$k = \begin{cases} n & | \quad n < m \\ m & | \quad \text{иначе} \end{cases}$$

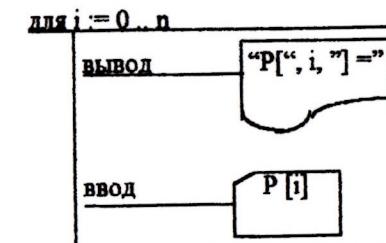
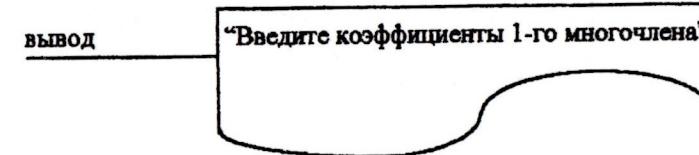
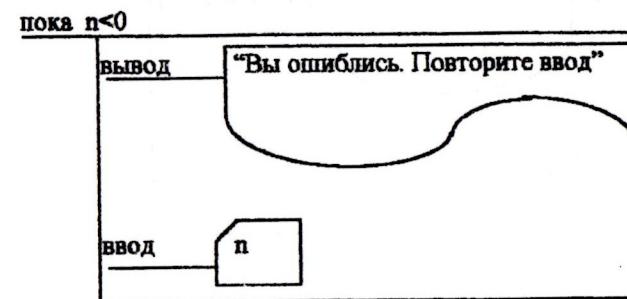
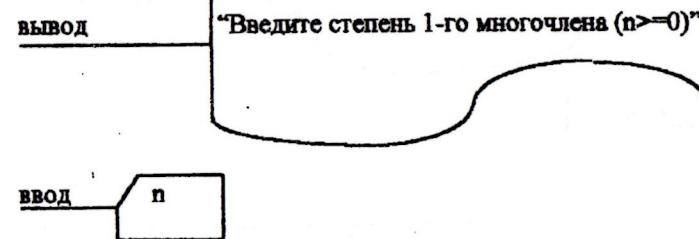
$$S = \begin{cases} n & | \quad n > m \\ m & | \quad \text{иначе} \end{cases}$$

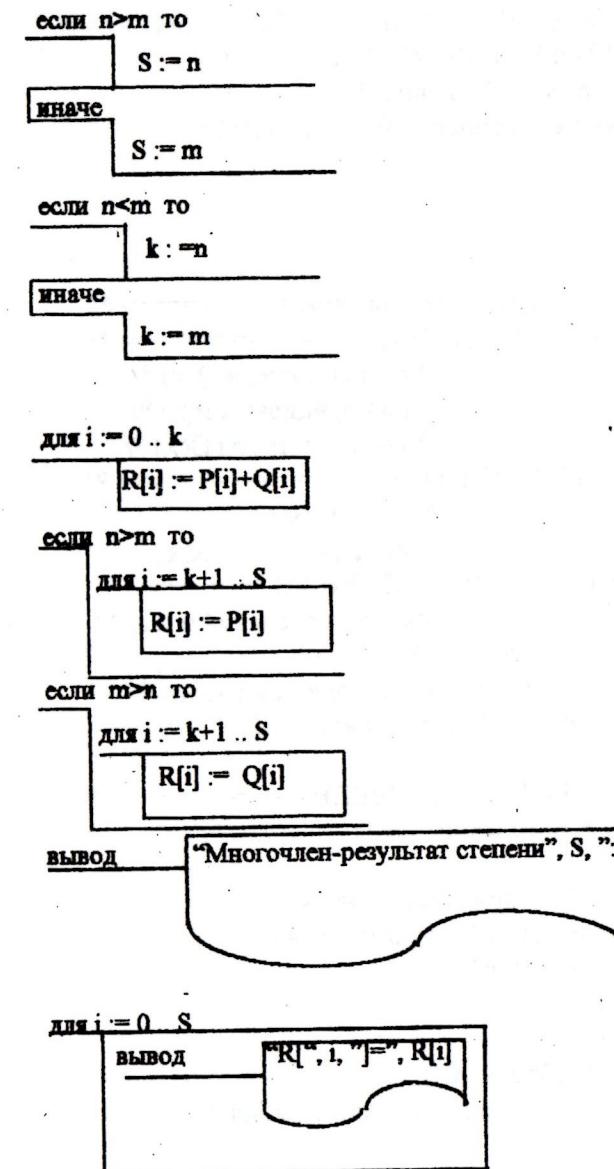
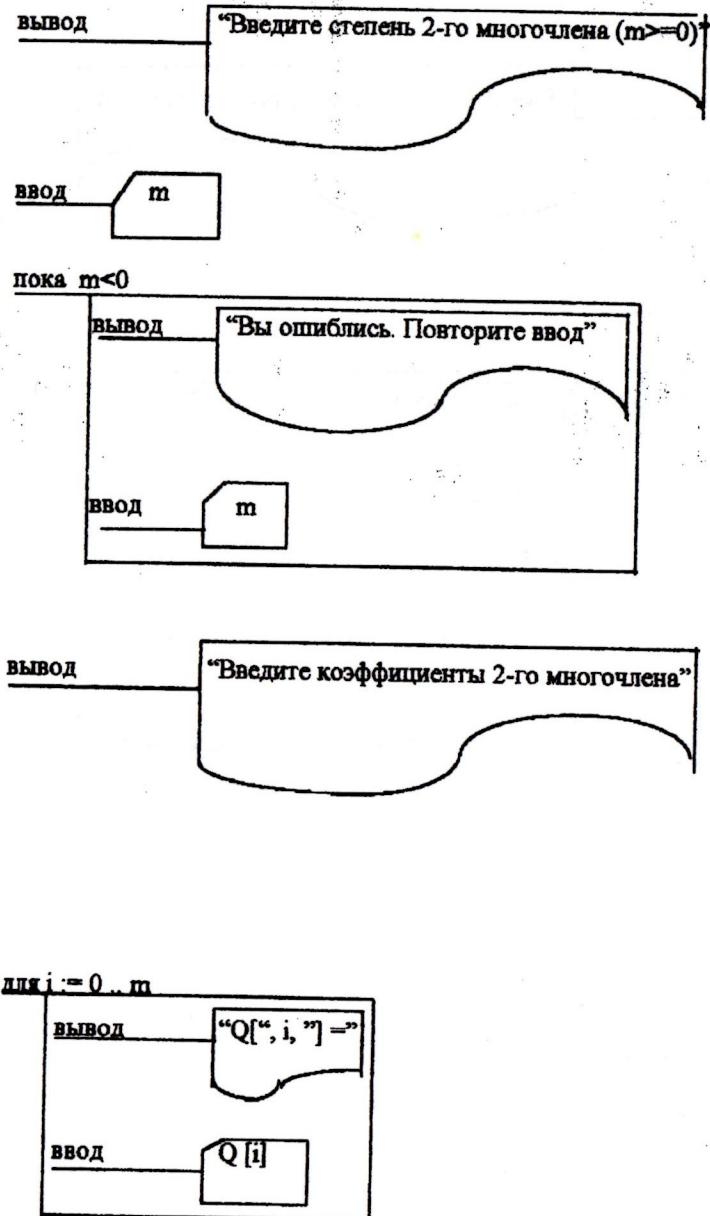
Промежуточные данные:

{минимальная из степеней многочленов}
{индекс}

k: целое
i: целое

4.1.4 Алгоритм





4.1.5 Текст программы на языке Си

```

/* = РАБОТА С МНОГОЧЛЕНАМИ И МАТРИЦАМИ == */
/*     ЛАБОРАТОРНАЯ РАБОТА N 3      */
/* студентки гр. МИ93 Ивановой С.    */
/* Найти сумму двух многочленов Pn(x) и Qm(x) */
#include <stdio.h>
#define N 100
#define M 100
main()
{
    /* ===== инструкции описания (декларации) ===== */
    /* ===== ИСХОДНЫЕ ДАННЫЕ ===== */
    int n, m;                                /* степени P(x) и Q(x) */
    float p[N+1],                             /* коэффициенты P(x) */
          q[M+1];                            /* коэффициенты Q(x) */
    /* ===== РЕЗУЛЬТАТЫ ===== */
    int s;                                    /* степень R(x) */
    float r[N+M+1];                          /* коэффициенты R(x) */
    /* ===== ПРОМЕЖУТОЧНЫЕ ДАННЫЕ ===== */
    int k,                                     /* наименьшая из степеней n и m */
        i;                                     /* индекс */
    int error;                                /* признак ошибки при вводе */
    /* =0, если нет ошибки, =1, если ошибка */

    /* ===== ПРОВЕРКА ОГРАНИЧЕНИЙ ===== */
    /* ===== для первого многочлена ===== */
    do
    {
        { error=0; /* обработка признака ошибки */
            printf("Введите степень 1-го многочлена");
            printf("(n>=0 и n<%d):\n", N);
            scanf("%d", &n);
            if( n<0 || n>N )
                { printf("Вы ошиблись.\n");
                  error=1; /* установка признака ошибки */
                }
        }
    } while (error);

}

```

```

/* ===== ввод 1-го многочлена ===== */
printf("Введите коэффициенты 1-го многочлена:\n");
for(i=0; i<n+1; i++)
{
    printf("\nP[%d]=", i);
    scanf("%f", &p[i]);
}

/* ===== для второго многочлена ===== */
do
{
    { error=0; /* обработка признака ошибки */
        printf("Введите степень 2-го многочлена");
        printf("(m>=0 и m<%d):\n", M);
        scanf("%d", &m);
        if( m<0 || m>M )
            { printf("Вы ошиблись.\n");
              error=1; /* установка признака ошибки */
            }
    }
} while(error);

/* ===== ввод 2-го многочлена ===== */
printf("\nВведите коэффициенты 2-го многочлена:\n");
for (i=0; i<m+1; i++)
{
    printf("\nQ[%d]=", i);
    scanf("%f", &q[i]);
}

/* ===== ИСПОЛНИМАЯ ЧАСТЬ ПРОГРАММЫ ===== */
/* ===== определение значений k и s ===== */
s = (n>m) ? n : m;
k = (n<m) ? n : m;
/* ===== формирование общей части ===== */
for(i=0; i<=k; i++)
    r[i] = p[i] + q[i];
/* ===== формирование оставшейся части ===== */
if (n>m)
    for(i=k+1; i<=s; i++)
        r[i] = p[i];

```

```

if (n<m)
for(i=k+1; i<=s; i++)
    r[i] = q [i];
/* ===== ВЫВОД РЕЗУЛЬТАТОВ ===== */
printf ("Многочлен-результат степени=%d\n", s);
for (i=0; i<s+1; i++)
    printf("r[%d]=%f\n", i, r[i]);
getch();
}

```

4.2 Приемы алгоритмизации и программирования при решении задач с матрицами

4.1.1 Представление математического объекта "матрица" и организация работы с матрицами

Матрица представляется двумерным массивом, причем в оперативной памяти он располагается по строкам:

matr: матрица [1..m, 1..n] вещественных.

Это описание матрицы из m строк и n столбцов.

Описание этой матрицы в языке Си:

```

#define M 15
#define N 10
float matr [M][N];           /* M строк и N столбцов */

```

Доступ к элементам матрицы осуществляется индексированием по номеру строки и номеру столбца: *matr [2,3]* - это 3-й элемент 2-й строки. В языке Си к этому же элементу доступ выглядит так: *matr [1][2]*.

4.2.2 Ввод данных в матрицу

При вводе данных в матрицу необходимо использовать два вложенных цикла. Организации ввода данных по строкам:

```

#define M 3
#define N 4
main{}
{float a [M][N];
 int i,j;
    /* 3 строки и 4 столбца */
    /* индексы строки и столбца*/

```

```

for (i=0; i<M; i++)
    for (j=0; j<N; j++)
        scanf ("%f", &a[i][j]);
}

```

Элементы такой матрицы A из входного потока

1.5	7.34	9.4	-11.2
5.5	4.6	-3.0	9.7
-2.25	5.6	6.6	0

должны быть введены в следующей порядке:

1.5 7.34 9.4 -11.2 5.5 4.6 -3.0 9.7 -2.25 5.6 6.6 0 .

4.2.3 Вывод данных матрицы

При выводе матрицы необходимо на устройстве вывода ее данные размещать по строкам. Например, матрица A из п.4.2.2 выводится следующим образом:

```

for (i=0; i<M; i++)
{ printf("%d-я строка: ", i+1);
    for (j=0; j<N; j++)
        printf(" %f ", a[i][j]);      /* вывод элементов строки */
    printf("\n");                      /* переход к новой строке */
}

```

На устройстве вывода эти данные будут размещены в следующем виде:

1-я строка: 1.5 7.34 9.4 -11.2

2-я строка: 5.5 4.6 -3.0 9.7

3-я строка: -2.25 5.6 6.6 0

4.2.4 Приемы алгоритмизации и программирования

Рассмотрим приемы алгоритмизации и программирования задач с матрицами на примере решения следующей задачи:

Найти k-ю степень вещественной матрицы порядка n.

Назовем исходную матрицу A, а матрицу-результат - C.

При решении этой задачи необходимо выделить три случая.

1) k=0. Результатом является единичная матрица, у которой на главной диагонали стоят единицы, а остальные элементы

равны 0. У элементов главной диагонали квадратной матрицы совпадают номера строки и столбца.

2) $k=1$. Содержимое исходной матрицы A необходимо скопировать в матрицу-результат C , что соответствует первой степени A .

3) $k>1$. При вычислении k -й степени матрицы следует использовать вспомогательную матрицу такого же порядка, что и исходная матрица (пусть это матрица B). В дальнейшем матрица C соответствует "следующей" (k -й) степени матрицы A , а матрица B - "предыдущей" ($k-1$)-й степени A . При переходе к следующей итерации повторяющегося процесса вычисления k -й степени матрицы A необходимо:

- скопировать содержимое матрицы C (k -й степени A) в матрицу B , сделав значение "следующей" степени матрицы A "предыдущей" степенью;

- найти "следующую" (k -ю) степень A умножением B на A :

$$A^k = A^{k-1} * A \quad (4.2)$$

Примечание: При нахождении произведения двух матриц используется формула

$$c_{ij} = \sum_{p=1}^n a_{ip} * b_{pj} \quad (4.3)$$

Приведем этапы решения этой задачи на ЭВМ.

4.2.5 Постановка задачи

4.2.5.1 Исходные данные

k : целое;	{ степень }
n : целое;	{ порядок матрицы }
a : массив [1..n, 1..n] вещественных;	{ матрица A }

4.2.5.2 Ограничения

$k \geq 0$

$n > 1$

4.2.5.3 Результаты

c : массив [1..n, 1..n] вещественных; { матрица A^k }

4.2.5.4 Связь

Если $k=0$, то результатом является единичная матрица ($C=E$).

Если $k=1$, то результатом является исходная матрица ($C=A$).

Если $k=1$, то результатом является матрица, вычисляемая рекуррентно: $C = A^n = A * A^{n-1}$.

4.2.6 Метод

$$c_{ij} = \begin{cases} 1 & | i=j \\ 0 & | \text{иначе} \end{cases} \quad j=\overline{1, n} \quad i=\overline{1, n} \quad k=0$$

$$c_{ij} = c_{ij} + a_{ip} * b_{pj} \quad p=\overline{1, n} \quad j=\overline{1, n} \quad i=\overline{1, n} \quad l=\overline{2, k} \quad k>1$$

$$b_{ij} = c_{ij} \quad j=\overline{1, n} \quad i=\overline{1, n} \quad l=\overline{2, k} \quad k>0$$

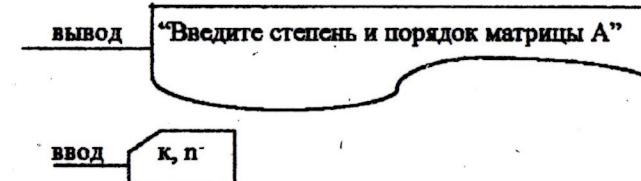
Промежуточные данные:

l : целое; { текущая степень }

i, j, p : целые; { индексы }

b : массив [1..n, 1..n] вещественных;
(матрица, соответствующая ($k-1$) степени A)

4.2.3 Алгоритм



пока $k<0$ или $n<-1$



вывод "Введите элементы матрицы A:"

для i := 1 .. n

для j := 1 .. n

вывод "Введите a [“i”, “j”]:"

ввод a [i, j]

если k > 0 то

для i := 1 .. n

для j := 1 .. n

c [i,j] := a [i,j]

если k > 1 то

для l := 2 .. k

для i := 1 .. n

для j := 1 .. n

b [i,j] := c [i,j]

для i := 1 .. n

для j := 1 .. n

c [i,j] := 0

для p := 1 .. n

c [i,j] := c [i,j] + a [i,p] * b [p,j]

иначе { иначе от "k>0" : случай k=0}

для i := 1 .. n

для j := 1 .. n

если i = j то

c [i,j] := 1

иначе

c [i,j] := 0

вывод k, "- ая степень матрицы A:"

для i := 1 .. n

для j := 1 .. n

вывод "c [“i”, “j”] =", c [i,j]

4.2.4 Текст программы на языке Си

/* == РАБОТА С МАТРИЦАМИ И МНОГОЧЛЕНАМИ == */

/* ЛАБОРАТОРНАЯ РАБОТА № 3 */

/* студента гр. МИ93 Семенова Р. */

/* Найти k-ю степень вещественной матрицы порядка n */

```
#include <stdio.h>
```

```
#include <conio.h>
```

```
#define N 50
```

```
main ()
```

```

{
/* ===== ИСХОДНЫЕ ДАННЫЕ ===== */
int k; /*степень*/
int n; /*порядок*/
float a[N][N]; /*матрица*/
/* ===== РЕЗУЛЬТАТЫ ===== */
float c[N][N];
/* ===== ПРОМЕЖУТОЧНЫЕ ДАННЫЕ ===== */
int i, j, p, l; /*индексы*/
float b[N][N]; /*(k-1) - я степень A*/
int error; /* код ошибки */
float r; /* рабочая переменная для ввода*/
/* ===== ИСПОЛНИМАЯ ЧАСТЬ ===== */
/* ----- проверка ограничений -----*/
do
{
    printf("Введите степень и порядок матрицы\n");
    error = 0;
    scanf("%d %d", &k, &n);
    if (k < 0 || n <= 1 || n > N)
        { printf("Ошибка, повторите ввод\n");
          error = 1;
        }
    }
while (error);
/* ----- ВВОД ЗНАЧЕНИЙ МАТРИЦЫ -----*/
printf("Введите элементы матрицы по строкам:\n");
for (i=0; i<n; i++)
    for (j=0; j<n; j++)
        { printf("\na[%d,%d]=", i+1, j+1);
          scanf("%f", &r);
          a[i][j]= r;
        }
/* ----- ВЫЧИСЛЕНИЕ k-Й СТЕПЕНИ МАТРИЦЫ -----*/
if(k>0) /* k>0 */
{ for (i=0; i<n; i++) /* это 1-я степень A */
    for (j=0; j<n; j++)
        c[i][j]=a[i][j];
}

```

```

/* ----- по степеням 2, 3 ... k ----- */
if (k>1)
{ for (l=2; l<=k; l++)
    /* ---- копия (k-1)-й степени в B ---- */
    { for (i=0; i<n; i++)
        for (j=0; j<n; j++)
            b[i][j]=c[i][j];
    /* ---- k-я степень A: ---- */
    for (i=0; i<n; i++)
        for (j=0; j<n; j++)
            {
                /* ---- очистка c[i][j] для k-й степени ---- */
                c[i][j]=0;
                /* ---- вычисление k-й степени ---- */
                for (p=0; p<n; p++)
                    c[i][j]+=(a[i][p]*b[p][j]);
            }
    /* цикл по j */
    }
    /* цикл по l */
    /* k>1 */
    /* k>0 */
    /* k=0 */
}
else
/* ----- формирование единичной матрицы ----- */
for (i=0; i<n; i++)
    for (j=0; j<n; j++)
        c[i][j]=(i==j)?1.0:0.0;
/* ----- ВЫВОД РЕЗУЛЬТАТА -----*/
printf ("%d-я степень матрицы A:\n", k);
/* ----- вывод C по строкам ----- */
for (i=0; i<n; i++)
{ for (j=0; j<n; j++)
    printf ("c[%d,%d]=%f; ", i+1, j+1, c[i][j]);
    printf ("\n");
}
getch();
} /* конец программы */

```

4.3 Варианты заданий

1. Для многочлена $P_n(x)$ получить его производную $P'_n(x)$ и вычислить значение $P'_n(2)$.
2. В квадратной матрице порядка n поменять местами столбец с номером k и строку с номером L .
3. Среди элементов, находящихся ниже главной диагонали квадратной матрицы, найти минимальный отрицательный элемент.
4. Определить, есть ли в прямоугольной матрице две строки из положительных элементов.
5. Даны две матрицы A и B . Найти $A^2 \cdot B^2$.
6. В прямоугольной матрице поменять местами второй и последний столбцы, состоящие только из отрицательных элементов.
7. Определить, является ли сумма элементов, находящихся на периметре прямоугольной матрицы, положительным числом.
8. Определить, является ли для заданных матриц A и B матрица $A^*B - B^*A$ единичной.
9. Данна прямоугольная матрица A . Переместить элементы ее периметра на один элемент по часовой стрелке.
10. В заданной прямоугольной матрице в последней строке найти первый отрицательный элемент. Все элементы столбца, соответствующего найденному отрицательному, умножить на минимальный элемент этого же столбца.
11. Данна квадратная матрица A порядка n и векторы X и Y того же размера. Получить вектор $A^*X + Y$.
12. Дан вектор Y размера n . Получить матрицу порядка $n \times m$ по следующему правилу: $a_{ij} = y_j^i$, $i \in [1, m]$, $j \in [1, n]$.
13. Определить, является ли отрицательным числом сумма элементов, стоящих в квадратной матрице выше побочной диагонали.
14. Найти норму прямоугольной матрицы A : $\| A \| = \min_j \sum_i |a_{ij}|$.
15. Определить, принадлежит ли максимальный элемент квадратной матрицы области, лежащей выше ее побочной диагонали.

16. В прямоугольной матрице определить местоположение максимального отрицательного элемента, модуль которого не больше заданного значения R .
17. Найти разность двух многочленов $P_n(x)$ и $Q_m(x)$ и вычислить значение этой разности при $x=3$.
18. Если в матрице найдутся две соседние строки, состоящие из четных элементов, то поменять их местами.
19. Для заданных матриц A и B найти $\frac{1}{2} * (A - B^2)$.
20. Данна квадратная матрица порядка n (n -четное). Является ли она симметричной, если ее сложить вдвое по вертикали?
21. Данна прямоугольная матрица A . Переместить элементы ее периметра на два элемента против часовой стрелки.
22. В квадратной матрице найти минимальный положительный элемент. Поменять местами строку и столбец, на пересечении которых расположен найденный минимум.
23. В квадратной матрице поменять местами элементы, симметричные относительно ее побочной диагонали.
24. Определить, является ли для заданных матриц A и B матрица $A^*B - B^*A$ нулевой.
25. Для многочлена $P_n(x)$ и числа q вычислить значение $q^*P_n(x)$ при $x=3.14$. Определить также количество отрицательных коэффициентов в многочлене-результате.
26. В прямоугольной матрице поменять местами первую и последнюю строки, содержащие хоть один нулевой элемент.
27. Определить, является ли произведение элементов, стоящих на главной и побочной диагоналях квадратной матрицы, отрицательным числом.
28. Определить, является ли для заданных матриц A и B матрица A^*B^2 положительно определенной.
29. Определить, сколько раз в прямоугольной матрице встречаются ненулевые элементы, а также местоположение максимального из них.
30. Для данного многочлена $P_n(x)$ найти значение $\int_a^b P_n(x) dx$ при заданных значениях a и b .
31. Определить, является ли произведение элементов, стоящих на периметре прямоугольной матрицы, отрицательным числом.

32. Дан вектор Y размера n . Получить матрицу A порядка $n \times m$ по следующему правилу: $a_{ij} = y_i^{j-1}$, $i \in [1, n]$, $j \in [1, m]$.
33. Определить, в какой части квадратной матрицы больше отрицательных элементов: лежащей выше или ниже главной диагонали.
34. Найти норму прямоугольной матрицы A : $\|A\| = \min_i \sum_j |a_{ij}|^2$
35. В прямоугольной матрице поменять местами первую и последнюю строки, состоящие только из нечетных элементов.
36. Есть ли в матрице строка и столбец, состоящие только из заданного значения?
37. Дописать в заданную прямоугольную матрицу последней строкой суммы соответствующих столбцов.
38. Если в заданном многочлене есть больше, чем k отрицательных коэффициентов, умножить его на заданное значение Z .
39. Определить, сколько на периметре заданной прямоугольной матрицы нулевых элементов.
40. Задана прямоугольная матрица. Переписать в матрицу результат тех ее строки, которые состоят из элементов, превышающих заданное значение s .

4.4 Контрольные вопросы по лабораторной работе

- 1) Организовать на языке Си ввод матрицы $A(m \times n)$ по столбцам.
- 2) Организовать на языке Си вычисление суммы элементов, стоящих на главной диагонали квадратной матрицы.
- 3) Выбрать структуру данных для представления математического объекта "разряженный многочлен", в котором часть степеней отсутствует, например:

$$P_{26}(x) = 2.57 - 3.76x^3 + 90.99x^7 - 8x^{26}$$

4.5 Содержание отчета

1. Условие задачи (использовать только арифметику с плавающей запятой, если это не оговорено особо).
2. Математические расчеты и вывод формул для раздела "Связь" постановки задачи.

3. Постановка задачи.
4. Метод решения задачи.
5. Алгоритм.
6. Текст программы на языке Си.
7. Таблица трассировки программы.
8. Графическая интерпретация тестовых примеров (с изображением формируемых многочленов, матриц и т.д.).
9. Выводы.
10. Программный документ "Описание программы".

5 ЛАБОРАТОРНАЯ РАБОТА N4. РАБОТА С СИМВОЛЬНЫМИ И ТЕКСТОВЫМИ ДАННЫМИ НА ЯЗЫКЕ СИ

Цель работы: освоить приемы алгоритмизации при работе с символьными и строковыми данными; использование метода “пошаговой детализации” и создание вспомогательных алгоритмов; определение функций пользователем и их вызов на языке Си; особенности программирования на языке Си при обработке текстов; использование функций стандартных библиотек языка Си при обработке символов и строк.

Контрольные вопросы:

- 1) Способы представления символьных данных в оперативной памяти.
- 2) Представление строковых констант в оперативной памяти.
- 3) Способы представления строковых переменных в оперативной памяти.
- 4) Способы описания строк в языке Си.
- 5) Операции для работы с символами в Си.
- 6) Средства для работы со строками в языке Си.
- 7) Определение пользователем функции и ее вызов в языке Си.
- 8) Различия в языке Си между функциями, возвращающими и не возвращающими значение результата.
- 9) Что такое “формальные” и “фактические” аргументы функций?

5.1 Приемы алгоритмизации и программирования при решении задач обработки текстов

5.1.1 Некоторые понятия, относящиеся к строкам (текстам)

Алфавит - это конечное множество символов.

Строка - это конечная последовательность символов некоторого алфавита. Пусть дан алфавит $A = \{0,1\}$. Строками в алфавите А являются: “0”, “111”, “01”, “10”, “00”, “101”, “0111” и т.д.

Пустая строка - это строка, не содержащая ни одного символа.

Важен порядок символов в строке: так, строки “01” и “10” различны.

Длина строки равна числу символов в строке. Таким образом, длина пустой строки равна нулю, длина строки “11” равна двум, длина строки “011” равна трем.

Если X и Y - строки, то их специалированием (или конкатенацией) является строка XY, полученная путем присоединения символов строки Y вслед за символами строки X. Например, конкатенация строк “01” и “10” равна “0110”.

Пусть X, Y и Z - произвольные (может быть и пустые) строки в некотором алфавите. Страна Z называется подстрокой (или инфиксом) строки S, если строка S представима как XYZ.

Подстрока X называется префиксом строки S, если строка S представима как XY.

Подстрока Y называется суффиксом строки S, если строка S представима как XY.

Строка X называется обратной к строке Y, если X представима как $A_1 A_2 \dots A_n$, а Y - как $A_n \dots A_2 A_1$, где A_i - символы.

Следующие примеры иллюстрируют введенные выше понятия:

- строка MAG - префикс строки MAGAZIN;
- строка TOMAT - суффикс строки AVTOMAT;
- строка PORT - подстрока (инфикс) строки ЭКСПОРТЕР;
- строка TORG - обратная к строке GROT.

Слово - это группы символов, разделенные пробелами (одним или несколькими) и не содержащие пробелов внутри себя.

Выравнивание текста влево заключается в перемещении слов к его левой границе, оставляя между последними ровно один пробел. Освободившиеся после такого перемещения символы текста (после последнего слова до конца текста) заполняются пробелами.

Выравнивание текста вправо заключается в перемещении слов к его правой границе, оставляя между последними ровно один пробел. Освободившиеся после такого перемещения символы текста (от начала текста до первого непробельного символа первого слова) заполняются пробелами.

5.1.2 Основные приемы работы с символами.

При работе с символами в языке Си важно помнить следующие

особенности:

1) символьная константа имеет тип int, а символьная переменная - тип char;

2) значение десятичной цифры не совпадает со значением соответствующей символьной константы:

$$1 \neq '1' \quad '9' \neq 9$$

3) значение символьной константы не совпадает со значением соответствующей строковой константы:

$$'1' \neq "1"$$

В оперативной памяти:

'1' - это

1

"1" - это

1	\0
---	----

Для символьного типа данных в языке Си определены все операции сравнения и сравнения на равенство.

Также в языке Си имеются специальные средства для анализа значения символа. Они представляются стандартной библиотекой, описанной в заголовочном файле *ctype.h*. Макросы этой библиотеки возвращают ненулевое значение (истину), если символ входит в состав заданного множества символов, и возвращают нуль (ложь), если он не входит в состав заданного множества. Все эти макросы являются таблично-управляемыми и очень быстрыми. Вы можете с помощью `#undef` отменить определение одного или нескольких имён этих макросов и запрограммировать их как обычные функции (не макросы).

Результат всех функций имеет тип int.

Ниже поясняется назначение каждого макроса:

int isalnum(int c) - с - алфавитно-цифровой символ (цифра или прописная или строчная буква);

int isalpha(int c) - с - алфавитный символ (прописная или строчная буква);

int isascii(int c) - с - символ ASCII (с от 0 до 0x7e);

int iscntrl(int c) - с - управляемый код ASCII (0x7f или от 0x00 до 0x1f);

int isdigit(int c) - с - цифра от '0' до '9';

int isgraph(int c) - с - печатный символ, кроме пробела (с от 0x21 до 0x7e);

int islower(int c) - с - строчная буква от 'a' до 'z';

int isprint(int c) - с - печатный символ или пробел (с от 0x20 до 0x7e);

int ispunct(int c) - с - знак пунктуации;

int isspace(int c) - с - "пробельный" символ: пробел, табуляция, возврат каретки, перевод строки, вертикальная табуляция или перевод формата;

int isupper(int c) - с - прописная буква от 'A' до 'Z';

int isxdigit(int c) - с - шестнадцатиричная цифра: от '0' до '9', от 'A' до 'F' или от 'a' до 'f'.

Таким образом, при анализе отдельного символа текста (строки) необходимо приводить аргументы перечисленных макросов к типу int. Например:

```
#include <stdio.h>
#include <ctype.h>
#define L 80
main()
{ char ss[L]; /* строка текста*/
  int i; /* порядковый номер символа в строке */
  for (i=0; i<L; i++)
    if (isdigit ((int) c[i]))
      printf ("Символ c[%d]=%c - это цифра\n", i, c[i]);
    else
      if (isalpha ((int) c[i]))
        printf ("Символ c[%d]=%c - это буква\n", i, c[i]);
}
```

5.1.3 Моделирование текстовых данных в языке Си

5.1.3.1 Моделирование отдельной строки

В языке Си нет типа данных "строка". Страна текста моделируется в языке одним из двух способов:

1) массив символов:

```
#define L 80
```

```
char str[L];
```



В этом случае можно в *str* разместить не более 79 символов, т.к. один символ необходим для хранения признака конца строки '\0'. Помните: для того, чтобы массив символов стал строкой в языке Си при посимвольном формировании его элементов, необходимо после последнего информационного символа строки записать признак конца строки - символьную константу '\0' ("нулевой символ").

Например, формируя строковое изображение константы π (3.14), в массив записываем 5 символов:

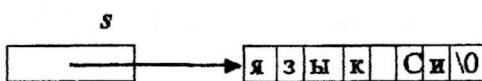
```
char s[5];
s[0]='3'; s[1]='.'; s[2]='1'; s[3]='4'; s[4]='\0';
```

2) указатель на символ:



В этом случае память под значение такой строки будет захватываться динамически. Например:

```
char *s = "язык Си";
```



Интенсивное использование таких строк (без явного динамического выделения памяти) делает программу нересурсоемкой.

5.1.3.1 Моделирование текста как массива строк

Массив строк может быть промоделирован в языке Си как двумерный символьный массив

```
#define N 80
#define L 64
char t[L][N]; /* текст из 64 строк по 80 символов в каждой*/
```

или как массив указателей на символ.

```
char* tt[L]; /* текст как массив из 64 указателей на символ*/
```

Массив строк, значения которых известны заранее, может быть проинициализирован в декларации:

```
char s1[3][]= {"январь", "февраль", "март"};
```

или

```
char *s2[3]= {"январь", "февраль", "март"};
```

Тогда доступ к элементу такого массива осуществляется так: *s1[1]* есть "февраль" (2-й месяц), а *s2[2]/0* есть "м" (1-я буква 3-го месяца).

5.1.4 Особенности работы со строками в языке Си

5.1.4.1 Особенности ввода строк в языке Си

Следует помнить, что при использовании форматированного ввода строк (функция *scanf* с символом преобразования %s):

- нет необходимости использовать оператор & ("взять адрес") перед именем вводимой строки (единственный случай, когда в функции *scanf* этот оператор не используется), например:

```
#define L 64
char str[L];
scanf("%s", str);
```

- ввод символов в строку в этом случае производится до первого встреченного пробела.

Для того, чтобы ввести строку, содержащую пробелы, следует использовать неформатированный ввод (функцию *gets*).

5.1.4.2 Особенности обработки строк в языке Си

Из-за отсутствия строкового типа данных в языке Си нет и операций для работы со строками. Все действия со строковыми данными обеспечиваются библиотекой *string.h*. Ниже приведено описание некоторых функций этой библиотеки.

- **STRCAT** - Конкатенировать строки.

Синтаксис *char *strcat (char *dest, const char *src);*

Описание Конкатенирует (объединяет) исходную строку *src* и

инициализированную результирующую строку *dest*, присоединяя последнюю к концу первой. Возвращает *dest*.

Параметры

char *dest - Указатель на инициализированную строку результата.

const char *src - Указатель на исходную строку, присоединяемую к концу результирующей.

- **STRCHR** - Искать в строке символ.

Синтаксис `char *strchr (const char *s, int c);`

Описание Ищет в строке *s* первое вхождение символа *c*, начиная с начала строки. В случае успеха возвращает указатель на найденный символ, иначе возвращает нуль.

Параметры

const char *s - Указатель на строку.

int c - Искомый символ. Для поиска нулевого символа, завершающего строку, укажите нулевое значение.

- **STRCMP** - Сравнить строки.

Синтаксис `int strcmp (const char *s1, const char *s2);`

Описание Сравнивает две строки. Возвращает отрицательное значение, если *s1*<*s2*; положительное значение, если *s1*>*s2*; нулевое значение, если *s1* и *s2* равны.

Параметры

const char *s1 - Указатель на первую сравнимую строку.

const char *s2 - Указатель на вторую сравнимую строку.

- **STPCPY** - Копировать строку в строку.

Синтаксис `char *strcpy (char *dest, const char *src);`

Описание Копирует исходную строку *src* и завершающий её нуль в строку результата *dest*, перезаписывая символы результирующей строки, расположенные в месте копирования. Возвращает *dest*.

Параметры

char *dest - Указатель на строку результата, перезаписываемую

исходной строкой. Результирующая строка не обязательно должна быть инициализированной.

const char *src - Указатель на завершающуюся нулевым символом исходную строку.

- **STRCSPN** - Найти подстроку, не содержащую заданных символов.

Синтаксис `int strcspn (const char *s1, const char *s2);`

Описание Возвращает длину максимальной начальной подстроки *s1*, не содержащей символов из второй строки *s2*. (Может использоваться для анализа элементов данных, например, для подсчёта начальных символов строки, которые не содержат некоторого знака пунктуации или расширения файла).

Параметры

const char *s1 - Стока, в которой ищутся символы, не содержащиеся в строке *s2*.

const char *s2 - Стока, содержащая символы, останавливающие процесс поиска символов в строке *s1*.

- **STRDUP** - Дублировать строку.

Синтаксис `char *strup (const char *s);`

Описание Коширует строку во вновь выделенный блок памяти длины *strlen(s)+1* байтов. Возвращает указатель на дублированную строку. Вы можете удалить эту строку, вызвав функцию *free()* с этим указателем в качестве аргумента. Если для запоминания дублированной строки не хватает памяти, функция возвращает нуль.

Параметры

const char *s - Указатель на дублируемую строку, завершающуюся нулём.

- **STRERROR** - Создать строку сообщения об ошибке.

Синтаксис `char *strerror (int errnum);`

Описание Возвращает указатель на строку сообщения об ошибке, определяемой параметром *errnum*. В конец строки

добавляет символ новой строки.

Параметры `int errnum` - Номер ошибки.

- **STRLEN** - Вычислить длину строки.

Синтаксис `int strlen (const char *s);`

Описание Возвращает длину строки *s* - число символов, предшествующих нулевому символу, завершающему строку.

Параметры `const char *s` - Указатель на завершающуюся нулем строку.

- **STRLWR** - Преобразовать в строке прописные буквы в строчные.

Синтаксис `char *strlwr (char *s);`

Описание Преобразовывает в строчные все прописные буквы в строке *s*.

Параметры `char *s` - Указатель на завершающуюся нулем строку.

- **STRNCAT** - Конкатенировать строки.

Синтаксис `char strncat(char *dest, const char *src, int maxlen);`

Описание Конкатенирует максимум *maxlen* символов исходной строки *src* к инициализированной результирующей строке *dest*.

Параметры

`char *dest` - Указатель на инициализированную строку результата.

`const char *src` - Указатель на исходную строку.

`int maxlen` - Максимальное число символов, копируемых из исходной строки в конец результирующей.

- **STRNCMP** - Сравнить части строк.

Синтаксис `int strncmp (const char *s1, const char *s2, int maxlen);`

Описание Аналогична функции *strcmp*, но сравнивает только максимум *maxlen* символов двух строк.

Параметры

`const char *s1` - Указатель на первую сравнимую строку.

`const char *s2` - Указатель на вторую сравнимую строку.

`int maxlen` - Максимальное число сравниваемых символов.

- **STRPBRK** - Искать символы в строке.

Синтаксис `char *strupbrk (const char *s1, const char *s2);`

Описание Ищет в строке *s1* первое вхождение любого символа из строки *s2*. Возвращает указатель на первый найденный символ, иначе (если символ не найден) возвращает нуль.

Параметры

`const char *s1` - Указатель на анализируемую строку.

`const char *s2` - Указатель на строку, содержащую множество символов для сравнения.

- **STRRCHR** - Искать в строке символ, начиная с конца строки.

Синтаксис `char *strrchr (const char *s, int c);`

Описание Эта функция аналогична *strchr*, но производит поиск последнего экземпляра (вхождения) данного символа *c* в строке *s*. Если символ найден, возвращает указатель на этот символ, иначе возвращает нуль.

Параметры

`const char *s` - Указатель на строку.

`int c` - Искомый символ.

- **STRREV** - Изменить порядок символов в строке на обратный.

Синтаксис `char *strrev (char *s);`

Описание Изменяет порядок следования символов в строке на обратный (кроме завершающего нулевого символа). Возвращает реверсированную строку *s*.

Параметры

`const char *s` - Указатель на строку.

- **STRSET** - Заменить все символы строки заданным символом

Синтаксис `char *strset (char *s, int c);`

Описание Заменяет все символы строки *s* заданным символом *c*, вплоть до завершающего строку нулевого байта, не включая его.

Параметры

const char *s - Указатель на инициализированную строку.
int c - Символ заполнения.

- **STRSPN** - Найти подстроку, содержащую заданные символы.

Синтаксис `int strspn (const char *s1, const char *s2);`

Описание Вычисляет длину максимальной начальной подстроки *s1*, содержащей только символы из строки *s2*. (Может использоваться при проверке полей данных, например, для подсчёта начальных цифр в строке).

Параметры

const char *s1 - Указатель на сканируемую строку.
const char *s2 - Указатель на строку, содержащую символы, которые ищутся в строке *s1*.

- **STRSTR** - Искать в строке подстроку.

Синтаксис `int strstr (const char *s1, const char *s2);`

Описание Ищет строку *s2* в другой строке *s1*. Возвращает адрес первого символа вхождения строки *s2* или, если подстрока *s2* не найдена в строке *s1*, - возвращает нуль.

Параметры

const char *s1 - Указатель на строку, в которой производится поиск подстроки *s2*.
const char *s2 - Указатель на строку, которая ищется в строке *s1*.

- **STRTOK** - Искать лексемы в строке.

Синтаксис `int strtok (char *s1, const char *s2);`

Описание Делит исходную строку *s1* на лексемы (подстроки), разделенные одним или несколькими символами-разделителями из строки *s2*. Если лексема найдена, возвращает ее адрес, иначе возвращает нуль. Обычно эта функция вызывается повторно. При первом вызове ей передается адрес исходной строки *s1*. Затем поиск следующих лексем продолжается до тех пор, пока функция не вернет нуль, при этом в вызове функции следует передавать нуль вместо аргумента *s1*.

Параметры

const char *s1 - Указатель на анализируемую строку. Эта функция изменяет исходную строку, заменяя символы-разделители нулями.

const char *s2 - Указатель на строку, содержащую один или несколько символов, используемых в качестве разделителей лексем.

- **STRUPR** - Преобразовать строчные буквы в прописные.

Синтаксис `char *strupr (char *s);`

Описание Преобразовывает в указанной строке все строчные буквы в прописные.

Параметры char *s - Указатель на завершающуюся нулём строку.

5.2 Пример

В исходном тексте есть изображения двузначных чисел. Переписать этот текст, заменяя найденное число его словесным эквивалентом.

Например, для текста

33 коровы, 33 коровы,
33 коровы - свежая строка.

результат будет

тридцать три коровы, тридцать три коровы,
тридцать три коровы - свежая строка.

5.2.1 Постановка задачи

5.2.1.1 Исходные данные

s1: строка [1..100]; {текст, содержащий двузначные числа}

5.2.1.2 Ограничения

длина (*s1*) ≠ 0 {строка *S1* - не пустая}

5.2.1.3 Результаты

s2 : строка [1..200]; {текст-результат}

сообщ : строка [1..60]; {строка-сообщение}

5.2.1.4 Связь

Копируем в строку *S2* посимвольно все элементы строки *S1*, кроме символов двузначного числа. Вместо символов

двузначного числа в S2 вставляем словесный эквивалент по следующим правилам:

- Если число принадлежит диапазону от 10 до 20, то словесный эквивалент состоит из одного числительного ("десять" ... "двадцать").
- В остальных случаях выбираем числительное для цифры десятков ("двадцать" ... "девяносто") и приписываем к нему числительное для цифры единиц ("один" ... "девять").

Дальнейшее копирование в S2 продолжаем с учетом вставки словесного эквивалента в S2 и пропуска двузначного числа в S1.

Если в S1 нет двузначного числа, то формируем строку-сообщение "В исходной строке нет двузначного числа".

5.2.2 Метод решения

5.2.2.1 Метод решения основной задачи

сообщ := "В исходной строке нет двузначного числа" | флаг ≠ 0

j := j+1
i := i+1
s2 := s1; | не (цифра (d1) и цифра (d2))
и d1 ≠ 0

i := i+2
j := j + длина (стр)
s2 := s2 || стр
стр := форм (d1, d2)
стр := ""
флаг := флаг + 1 | цифра (d1) и цифра (d2)
и d1='0'

d2 := s1[i+1]
d1 := s1[i]
стр := ""
флаг := 0 | пока
i ≤ длина(s1)

j := 1
i := 1

Промежуточные данные:

i, { порядковый номер символа в исходной строке}
j, { порядковый номер символа в выходной строке}
флаг: целые; {количество двухзначных чисел в исходной строке }
d1, { символ первой цифры двухзначного числа}
d2: символы; { символ второй цифры двухзначного числа }
стр: строка [1..100]; {строка, содержащая словесный эквивалент
для чисел}

5.2.2.2 Метод решения вспомогательной подзадачи форм(d1, d2)

возврат (рез)

рез := рез || назв_единиц id2 | d2 ≠ 0

рез := назв_десятки id1 | d1 ≠ 1

возврат (назв11_19 id2) | d2 ≠ 0

возврат (назв_десятки 1) | d2 = 0

id2 := d2 - '0'
id1 := d1 - '0'

Промежуточные данные:
id1, {число, соответствующее 1-й цифре числа}
id2: целые; {число, соответствующее 2-й цифре числа}
рез: строка [1..100]; {строка для формирования словесного
представления чисел от 20 до 99}
назв единиц: массив [1..9] строк;

один	два	три	четыре	пять	шесть	семь	восемь	девять
------	-----	-----	--------	------	-------	------	--------	--------

назв11_19:массив [1..9]строк; назв_десятков:массив[1..9] строк;

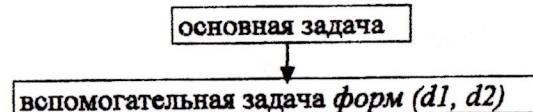
одиннадцать
двенадцать
тринадцать
четырнадцать
пятнадцать
шестнадцать
семнадцать
восемнадцать
девятнадцать

десять
двадцать
тридцать
сорок
пятьдесят
шестьдесят
семьдесят
восемьдесят
девяносто

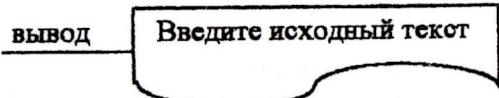
Параметры подзадачи:

d1, { символ первой цифры двухзначного числа}
d2: символы; { символ второй цифры двухзначного числа }

5.2.2.3 Иерархия основной задачи и вспомогательной подзадачи



5.2.3 Алгоритм основной программы



i := 1
j := 1
флаг := 0
копировать (s2, "")

пока (i < длина (s1))

d1 := s1[i]
d2 := s1[i+1]
если цифра(d1) и цифра (d2) и d1 ≠ 0 {символы числа}

флаг := флаг+1
копирование (стр, "")
копирование (стр, форм (d1,d2))
конкатенация (s2, стр)
j := j+длина (стр)
i := i+2

иначе {все прочие символы}

s2[j] := s1[i]
i := i+1
j := j+1

если флаг ≠ 0

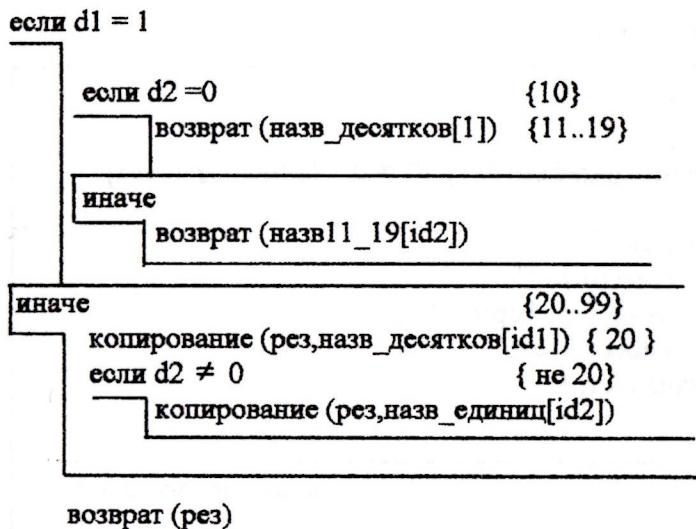
вывод s2

иначе

вывод "В", s1, "нет двузначного числа"

5.2.3 Вспомогательный алгоритм

id1 := d1 - '0'
id2 := d2 - '0'



5.2.4. Текст программы на языке Си

```

/* = РАБОТА С СИМВОЛЬНЫМИ И ТЕКСТОВЫМИ ДАННЫМИ */
/*
    ЛАБОРАТОРНАЯ РАБОТА N 4
*/
/*
    студентки гр. МИ93 Ивановой С.
*/
/* В исходном тексте есть изображения двузначных чисел. */
/* Переписать этот текст, заменяя найденное число его */
/* словесным эквивалентом. */
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
#include<conio.h>
#define L 300
#define LF 150
/* ПРОТОТИП ФУНКЦИИ, ОПРЕДЕЛЕННОЙ ПОЛЬЗОВАТЕЛЕМ */
char *form(char d1, char d2);
main()
{
/* ===== ИСХОДНЫЕ ДАННЫЕ ===== */
char s1[L];          /* исходная строка */

```

```

/* ===== ВЫХОДНЫЕ ДАННЫЕ ===== */
char s2[L+LF];        /* выходная строка */
/* ===== ПРОМЕЖУТОЧНЫЕ ДАННЫЕ ===== */
char ss[2]; /* вспомогательная строка для копирования s1 в s2*/
char str[LF]; /* промежуточная строка для формирования */
                /* словесного представления числа */
char d1,           /* символьное представление первой цифры числа */
d2;               /* символьное представление второй цифры числа */
int i=0;            /* порядковый номер текущего символа во входной*/
                    /* строке*/
int j=0;            /* порядковый номер текущего символа в выходной*/
                    /* строке*/
int fl=0;           /* признак наличия двузначного числа в строке s1: */
                    /* не 0 - есть, 0 - нет */
/* ===== ИСПОЛНИМАЯ ЧАСТЬ АЛГОРИТМА ===== */
clrscr();
printf("Введите исходный текст (до %d символов):\n", L);
gets(s1);
printf("\nИсходный текст:\n");
puts(s1);
strcpy(s2,"");
while(i<strlen(s1))           /* не конец s1 */
{
    d1=s1[i];                  /* 1-я цифра числа */
    d2=s1[i+1];                /* 2-я цифра числа */
    if( isdigit((int)d1) && isdigit((int)d2) /* d1 и d2 - цифры ? */
        && d1 != '0')           /* 1-я цифра не 0 ? */
        {
            fl++;                /* найденное в s1 слово - 2-значное число */
            strcpy(str,""); /* в str - словесное представление числа */
            strcpy(str,form(d1,d2));
            strcat(s2,str); /* дописывание в s2 строки str */
        }
    /* смещаем порядковый номер символа в выходной строке */
    j+=strlen(str);
    /* смещаем порядковый номер символа во входной строке */
    i+=2;
}
else                                /* в s1 нет двух цифр подряд*/

```

```

{
/*переписываем один символ и перемещаем порядковый номер */
/* в строках s1 и s2: */
    ss[0]=s1[i++];
    ss[1]='\0';
    strcat(s2, ss);
    j++;
}
/* ===== ВЫВОД РЕЗУЛЬТАТОВ ===== */
printf("\n результат:\n");
if (fl) /* есть 2-значное число */
    puts(s2);
else
    printf("В %s нет двузначного числа.\n",s1);
getch();
return;
}
/* ===== ВСПОМОГАТЕЛЬНЫЙ АЛГОРИТМ ===== */
char *form (char d1, char d2)
/* ===== ПРОМЕЖУТОЧНЫЕ ДАННЫЕ ===== */
char res[LF]; /*строка в которой формируется результат функции*/
/* названия единиц: */
char * ed[] = { "один", "два", "три", "четыре", "пять",
                "шесть", "семь", "восемь", "девять"};
/* названия десятков: */
char *dec[] = { "десять", "двадцать", "тридцать",
                "сорок", "пятьдесят", "шестьдесят",
                "семьдесят", "восемьдесят", "девяносто"};
/* названия чисел от 11 до 19: */
char *ch11_19[] = {
    "одиннадцать", "двенадцать", "тринадцать",
    "четырнадцать", "пятнадцать", "шестнадцать",
    "семнадцать", "восемнадцать", "девятнадцать"};
int id1=d1-'0'; /* числовое представление 1-й цифры числа */
int id2=d2-'0'; /* числовое представление 2-й цифры числа */

```

```

/* ===== ИСПОЛНИМАЯ ЧАСТЬ АЛГОРИТМА ===== */
if (id1==1) /* это числа 10..19 */
    if (id2==0) /* это 10 */
        return dec[0];
    else
        /* одиннадцать .. девятнадцать */
        return ch11_19[id2-1];
else /* это числа 20..99 */
    /* копирование слова, соответствующего десяткам */
    strcpy(res, dec[id1-1]);
    /* добавление слов, соответствующих единицам */
    if (d2 != 0) /* это числа 21..99 */
        strcat(res, ed[id2-1]);
    }
    return(res);
}

```

5.3 Варианты заданий

Во всех вариантах заданий дан текст. Группы символов, разделенные пробелами (одним или несколькими) и не содержащие пробелов внутри себя, будем называть, как и прежде словами.

1. В тексте найти самое длинное слово, содержащее изображение десятичного числа.
2. Текст представляет собой программу на языке Си. Подсчитать количество строк в фрагменте программы между определением некоторого макроса (команда препроцессора `#define <имя_макроса> <определение_макроса>`) до отмены этого макроопределения (команда препроцессора `#undef <имя_макроса>`).
3. Из исходного текста в выходной переписать все слова, которые являются изображениями шестнадцатеричных чисел.
4. Найти в тексте все пары слов-соседей, второе из которых начинается на ту же букву, которой заканчивается первое из такой пары.

5. Сколько в заданном тексте имен собственных ?
6. Найти первое вхождение в тексте слова, начинающегося с заданной приставки.
7. В тексте слова отделены друг от друга любым количеством пробелов. Сформировать выходную строку, в которой содержатся первые буквы слов исходного текста.
8. Определить длину самого последнего палиндрома текста.
9. Даны строки S. Построить строки S1 и S2 из символов строки S, стоящих на четных (S1) и нечетных (S2) местах соответственно.
10. Найти в тексте все слова со спаренными согласными (например, это слова rассвет, звуковой, искусственный, accommodation, add).
11. Найти в тексте самое длинное слово, содержащее спаренные гласные (например, это слова поохотиться, заарканить, переезд, need).
12. Преобразовать исходный текст следующим образом: перед всеми глаголами в инфинитиве (оканчивающихся на -ить, -ать, -ять, -еть) вставить отрицание "не".
13. Определить, каких слов в тексте больше: состоящих только из букв или только из цифр ?
14. Текст представляет собой программу на языке Си. Удалить из него все комментарии (текст, заключенный между скобками /* и */, включая эти символы).
15. Текст представляет собой программу на языке Си. Подсчитать количество инструкций *for*, вложенных друг в друга.
16. Текст представляет собой программный документ, подготовленный согласно ГОСТу. Сформировать оглавление документа, в котором содержатся названия глав этого документа.
17. Текст предоставляет собой арифметическое выражение. Проверить согласованность круглых скобок в этом выражении.
18. Текст представляет собой программу на языке Си. Подсчитать максимальную глубину вложенности операторных скобок в этой программе.
19. Сколько в данном тексте слов с ошибочным употреблением частиц *-то*, *-либо*, *-нибудь*, *-кое* (без символа '-') ?
20. В англоязычном тексте изменить все окончания слов *-ed* на окончания *-ing*.
21. Каких артиклей в данном англоязычном тексте больше - определенных или неопределенных ? (неопределенный артикль "*the*", определенный - "*a*", "*an*").
22. Чего в исходном тексте больше: заданных приставок или совпадающих с ними предлогов ? Примеры таких приставок и предлогов: *за*, *в*, *с*, *по* .
23. Текст представляет собой программу на языке Си. Подсчитать, сколько раз в нем встречаются именованные константы.
24. Даны строки S1 и S2. Проверить, является ли строка S1 префиксом S2.
25. Определить, есть ли в данной строке слово, являющееся изображением числа, кратного 5.
26. В исходной строке содержится N предложений, каждое из которых заканчивается точкой, вопросительным или восклицательным знаком. Переписать этот текст так, чтобы в выходной строке каждое предложение начиналось с "красной" строки. Длина строки выходного текста равна M, а величина абзацного отступа равна трем.
27. Найти все вхождения в тексте слов, заканчивающихся заданным окончанием.
28. Даны строки S1 и S2. Проверить, является ли строка S1 суффиксом S2.
29. Найти в заданном тексте максимальную по длине подстроку, состоящую только из цифр. Сообщить начальный и конечный номера символов найденной подстроки и ее длину.
30. Текст представляет собой программу на языке Си. Заменить в нем все комментарии языка Си на соответствующие комментарии языка Паскаль ("/*" - на "{", "*/" - на "}").
31. Дан англоязычный текст, состоящий из утвердительных предложений, заканчивающихся точкой. Превратить все предложения в вопросительные (в форме общего вопроса). Например, предложение "My name is Victor." становится "My name is Victor, yes?"

32. Дан русскоязычный текст, в котором встречаются сокращения. Заменить в этом тексте все сокращения на их полный текст. Допустимые сокращения: *т.к.* (так как), *т.н.* (так называемый), *др.* (другие).
33. Выполнить выравнивание исходной строки длиной N по правому краю. Длина строки-результата равна M ($M > N$).
34. Сколько в тексте слов, являющихся изображениями вещественных чисел с экспоненциальной частью?
35. В тексте слова разделены друг от друга любым количеством пробелов. Сформировать выходную строку, в которой содержатся последние буквы слов исходного текста.
36. Дан текст из печатных и управляющих символов. Известно, что он зашифрован некоторым кодом: каждый *n*-й символ является "лишним". Расшифровать этот текст.
37. Известны две нотации для почтового адреса физического лица:

старая нотация:

Страна

Индекс

Город

Улица, дом

Ф.И.О

новая нотация:

Ф.И.О

Дом, улица

Город

Индекс

Страна

Составить программу-конвертор, которая приводит адрес в старой нотации к новой.

38. Напечатать для заданного месяца календарь, расположив дни недели по горизонтали в виде:

пн вт ср чт пн сб вс.

39. Найти в заданном тексте минимальную по длине подстроку, состоящую только из символов, приравненных к буквам в алгоритмических языках.

40. Задан текст песни. Определить, повторяется ли в этой записи полный текст припева, или он приведен ровно один раз. Текст припева отделяется от куплета соответствующим заголовком.

5.4 Контрольные вопросы по лабораторной работе

1) Назначение прототипа функции в языке Си.

- 2) Особенности передачи аргументом функции данных символьного и строкового типа в языке Си.
- 3) Реализовать на языке Си библиотечную функцию *strcpy*.
- 4) Реализовать на языке Си библиотечную функцию *strcmp*.
- 5) Реализовать алгоритм подсчета слов текста, состоящих только из символов в верхнем регистре.
- 6) Организовать на языке Си ввод текста произвольной длины и вывод его по 64 символа в строке.
- 7) Упорядочить на языке Си исходный текст по возрастанию размера его строк. Подобрать наиболее эффективный способ представления исходного текста. Сортировку выполнить без использования вспомогательного массива строк.
- 8) На языке Си в заданном арифметическом выражении выделить все имена переменных и числовые константы с помощью функции *strtok*. Какой должна быть в общем случае строка разделителей, используемая названной функцией?

5.5 Содержание отчета

1. Условие задачи.
2. Постановка задачи.
3. Методы решения задачи и вспомогательных подзадач; иерархия подзадач.
4. Основной и вспомогательные алгоритмы.
5. Текст программы на языке Си. Если задача может быть решена с использованием только библиотечных функций, такой альтернативный вариант решения представить для сравнения результатов работы.
6. Таблица трассировки программы.
7. Графическая интерпретация тестовых примеров (с изображением формируемых строк, указателей их занятости, счетчиков и т.д.).
8. Выводы.
9. Программный документ "Описание программы".

6 ЛАБОРАТОРНАЯ РАБОТА N5. РАБОТА С ТАБЛИЧНЫМИ ДАННЫМИ НА ЯЗЫКЕ СИ

Цель работы: освоить приемы алгоритмизации при организации обработки табличных данных.

Контрольные вопросы:

- 1) Тип данных в языках программирования “запись”.
- 2) Тип данных в языках программирования “таблица”.
- 3) Назовите способы задания в языке Си *структурного типа* данных.
- 4) Назовите способы объявления в языке Си *данных структурного типа*.
- 5) Как выполняется инициализация данных структурного типа в языке Си ?

6.1 Приемы алгоритмизации и программирования при решении задач обработки табличных данных

6.1.1 Объявление данных типа “структура”

При решении данного класса задач объявление типа данных “структура” предпочтительнее выполнять либо с использованием тега структурного типа, либо с помощью конструкции *typedef*. Такой подход позволяет использовать имя структурного типа как при объявлении собственно табличных данных, так и при объявлении промежуточных данных того же типа, используемых при обработке таблиц. Например, объявление исходной таблицы для примера 6.2 с использованием тега структурного типа:

```
#define N 30
#define LN 20
#define LC 15
struct country /* структурный тип с тегом country/
{
    char name[LN+1];
    char capit[LC+1];
    float area;
    float chisl;
};
```

```
struct country c[N]; /* таблица типа struct country */
struct country rab; /* рабочая переменная этого же типа */
```

Совмещение объявления структурного типа с декларацией данных следует использовать в том случае, когда требуется объявить только один объект программы этого типа. Например, объявление таблицы-результата для примера 6.2 без использования тега структурного типа:

```
struct /* описание таблицы-результата */
{
    char name[LN+1];
    char capit[LC+1];
    float pl;
} c_r[N];
```

При объявлении членов структуры, представляющих собой строку, резервируем дополнительный байт для признака конца строки: *название страны* (*name*) - это строка из LN символов + 1 байт на нулевой символ.

6.1.2 Ввод-вывод данных типа “структура”

При организации ввода-вывода данных типа “структура” необходимо помнить следующее.

- 1) Данные типа “структура” могут быть введены (выведены) только поэлементно.
- 2) Если при этом используются функции форматированного ввода-вывода, то каждый член структуры имеет свой собственный символ преобразования в форматной строке.

6.1.2.1 Особенности ввода данных типа “структура”

При вводе каждого из членов структуры пользователю необходимо сообщить в виде приглашения семантику (наименование) соответствующего поля. Для обеспечения “дружественного” пользователю интерфейса рекомендуется организовать ввод данных, используя разметку в виде “шапки” таблицы с учетом размера членов структуры.

Рассмотрим приемы организации ввода данных в исходную таблицу примера 6.2. Для горизонтального отчеркивания в

таблицах используем символ ‘-’, а для вертикального (разделения столбцов таблицы) - символ ‘!’. С учетом количества символов, отведенных для каждого члена структуры типа *struct country*, размещаем наименования соответствующих членов структуры в шапке таблицы. Эти наименования столбцов таблицы центрируем, добавляя приблизительно равное количество символов пробела слева и справа от текста (рекомендуется предварительно макет шапки таблицы подготовить на листе в клетку). С учетом необходимого количества символов-разделителей столбцов определяем суммарную длину шапки таблицы по горизонтали:

семантика члена структуры	размер члена структуры (байты)	размер наименования столбца (байты)
название страны	20	название страны (15)
название столицы	15	столица (7)
площадь территории	произвольный	площадь (7)
численность населения	произвольный	численность (11)

Горизонтальный размер шапки таблицы равен

$$\underbrace{20 + 15}_{\substack{\text{k1 членов} \\ \text{типа } \underline{\text{char}}}} + \underbrace{7 + 11}_{\substack{\text{k2 членов} \\ \text{типа } \underline{\text{float}}}} + \underbrace{k1 + k2*2 + (k2-1)*3}_{\substack{\text{!} \\ \text{!} \\ \text{!} \\ \text{!}}} = 62$$

$k1=2, k2=2$

Теперь получаем макет шапки таблицы:

!--название-страны--!---столица---!-площадь-!-численность-

Так как строка горизонтального отчеркивания в таблице используется неоднократно, ее целесообразно оформить в программе на языке Си как переменную, которая проинициализирована значением соответствующей константы:

```
#define LG 62
char sg[LG+1] = "-----";
```

и затем используется как аргумент функции вывода:

```
/* печать шапки таблицы*/
printf("Введите данные о странах\n");
printf("%s\n", sg);
printf("! название страны ! столица !");
printf("площадь ! численность !\n");
printf("%s\n", sg);
```

Вывод на экран подобного приглашения наглядно демонстрирует пользователю размеры полей вводимых членов структуры. Ввод данных в этом случае производится следующим образом:

- в соответствующем столбце таблицы вводится значение члена структуры, причем друг от друга они должны быть разделены пробелом (-ами);
- ввод отдельной структуры (строки таблицы) завершается нажатием клавиши *ENTER*.

Пример ввода для рассмотренной таблицы:

Введите данные о странах:

!--название-страны--!---столица---!-площадь-!-численность-

Франция	Париж	0.544	57.529	„
Аргентина	Буэнос-Айрес	2.780	33.106	„

В данном протоколе сеанса ввода данных для наглядности символы разделителя-пробела изображены знаком ‘-’, а нажатие клавиши *ENTER* - символом ‘„’.

6.1.2.2 Особенности вывода данных типа “структура”

При выводе данных типа “структурка” (“таблица”) также необходимо предварительно рассчитать макет шапки. Кроме этого, для обеспечения вывода данных в столбцы заданной ширины необходимо воспользоваться такими элементами форматной строки как “ширина поля” и “выравнивание” (для строк) и “точность” (для данных с плавающей точкой).

Так строка таблицы-результата примера 6.2 выводится:

```
printf("%-20s%-15s% 5.2f\n", c_r[i].name, c_r[i].capit, c_r[i].pl);
```

Пример вывода для рассмотренной таблицы:

—название страны—	—столица—	—плотность—
Франция	Париж	0.11
Аргентина	Буэнос-Айрес	0.01

6.1.3 Проверка ограничений при вводе данных типа "таблица"

Структурированное данное типа "таблица" вводится поэлементно (построчно), как и в случае данного типа "массив".

Значения членов структуры при вводе табличных данных следует обязательно проверять на корректность. Поэтому при вводе данных в таблицу необходимо идентифицировать порядковый номер строки таблицы и корректировать его значение в программе на языке Си. Использование цикла for в Си позволяет это сделать естественным образом, так как переменная цикла может быть изменена в теле for:

```
... do {                                /* для примера 6.2 */
    for (i=0; i<n; i++)
        { printf("%d ", i+1); /* идентификация номера строки */
        scanf("%s", c[i].name);  scanf("%s", c[i].capit);
        scanf("%f", &r);       c[i].area=r;
        scanf("%f", &r);       c[i].chisl=r;
        if(c[i].area<0 || c[i].chisl<0) /* проверка ограничений*/
            {pr=1;
             printf("Вы ошиблись! Повторите ввод\n");
             i--;      /* корректировка номера строки таблицы */
            }
        else pr=0;
    }
} while(pr);
```

Этот же фрагмент программы напоминает о необходимости использования вспомогательной переменной (в данном случае *r*) при вводе членов структуры, имеющих тип *float*.

6.1.4 Вопросы эффективности работы программы при обработке табличных данных

6.1.4.1 Выбор и эффективное использование промежуточных переменных

При решении таких распространенных задач для табличных данных, как нахождение строки, соответствующей минимуму или максимуму значения некоторого столбца, необходимо использовать рабочую переменную для хранения значения текущего минимума (максимума). Возможны два способа реализации такой рабочей переменной.

Способ 1. Рабочая переменная имеет тот же тип данных, что и элемент соответствующей таблицы. Член структуры, который соответствует искомому минимуму (максимуму), участвует в сравнении; в остальные члены копируются значения строки таблицы для хранения этого текущего минимума (максимума). После завершения просмотра таблицы в соответствующем члене рабочей переменной хранится искомый результат. Для примера 6.2 этот способ решения выглядит так:

```
struct country rab;      /* переменная для текущего минимума */
rab.area=1.0e6;           /* площадь, превышающая все известные */
for(i=0; i<n; i++)
    if (c[i].area<rab.area)
        /* запоминание в rab всей строки таблицы*/
        strcpy(rab.name, c[i].name);
        strcpy(rab.capit, c[i].capit);
        rab.area= c[i].area;
        rab.chisl= c[i].chisl;
    }
/*===== вывод результата =====*/
printf("Страна с минимальной площадью населения %s",
       rab.name);
```

Способ 2. Рабочая переменная имеет тот же тип данных, что и искомый минимум (максимум). Эта переменная участвует в сравнении. Во второй рабочей переменной запоминается только номер строки таблицы, соответствующий искомому минимуму (максимуму), вместо копирования всех членов структуры из

строки таблицы. После завершения просмотра таблицы в первой рабочей переменной хранится искомый минимум (максимум). Информацию о прочих членах структуры получают путем доступа к элементу таблицы по номеру, хранящемуся во второй рабочей переменной. Для примера 6.2 этот способ решения выглядит так:

```
float ar_min; /* переменная для текущего минимума */
int n_min; /* номер элемента таблицы с текущим минимумом */
n_min=-1;
ar_min=1.0e6; /* площадь, превышающая все известные */
for(i=0; i<n; i++)
    if(c[i].area<ar_min)
        {ar_min=c[i].area;
        n_min=i;
        }
/*===== вывод результата =====*/
printf("Страна с минимальной площадью населения %s",
       c[n_min].name);
```

6.1.4.2 Исключение повторных просмотров таблицы

Если исходная задача предусматривает решение нескольких подзадач, использующих перебор всех ее элементов (полный просмотр таблицы), то для повышения эффективности программы совмещают их решение в одном просмотре. Такими подзадачами являются:

- выбор элементов таблицы, удовлетворяющих некоторому условию или или их комбинации;
- нахождение элемента (-тов) таблицы, соответствующих минимальному (максимальному) значению по некоторому столбцу.

Преимущество такого совмещения заключается в сокращении времени непосредственно перебора (что важно при больших размерах таблиц), а также в уменьшении размера кода программы. Решение примера 6.2 выполнено с учетом данного приема программирования.

6.2 Пример

Дана информация о странах мира в виде: название, столица, площадь (млн. кв. км) и численность населения (млн. чел.). Сформировать список тех стран, плотность населения которых превышает заданную. Об этих странах сообщить их название, столицу и плотность населения. Найти страну с минимальной площадью территории.

6.2.1 Постановка задачи

6.2.1.1. Исходные данные

```
n,                                     { количество стран }
дл_назв,                                { размер строки названия страны }
дл_ст: целые;                          { размер строки названия столицы страны }
страны: таблица [1..n]
    ( название: строка [1..дл_назв];      { название }
      столица: строка [1..дл_ст];        { столица }
      плош: вещественное;                 { площадь }
      числ: вещественное                 { численность }
    );
пл_задан: вещественное;                { заданная плотность населения }
```

6.2.1.2. Ограничения

```
n>0
дл_назв>0;   дл_ст>0
страны;назв≠''; страны;стол≠'';           | i=1..n
страны;плош>0;  страны;числ>0;
пл_задан>0
```

6.2.1.3 Результаты

```
стр_p: таблица [1..n]
    ( название: строка [1.. дл_назв];      { название }
      столица: строка [1.. дл_ст];        { столица }
      пл: вещественное                     { плотность населения }
    );
р: целое;          { реальное количество стран в таблице-результате }
назв_мин_плош: строка [1.. дл_назв]; { название страны с
                                         минимальной площадью территории }
сообщ: строка [1..60];               { строка-сообщение}
```

6.2.1.4 Связь

```

Эр (стр_pp := стрi;   стрi.числ / стрi.площ > пл_задан);
      | i ∈ [1..n];   p ∈ [1..n];
Э назв_мин_площ (назв_мин_площ := min (стрi.площ;   i ∈ [1..n])

```

6.2.2. Метод решения

сообщ := 'Стран с площадью населения, превышающей заданную не найдено' | p=0

назв_мин_площ := стр_{нмин}.назв

{ поиск страны с минимальной площадью территории }

нмин := i

шлмин := стр_i.площ | стр_i.площ < шлмин | i = 1..n

шлмин := 1000000

нмин := 0

{ поиск стран с плотностью населения, превышающей заданную }

стр_p_p.пл := пл

стр_p_p.столица := страны_i.столица

стр_p_p.назв := страны_i.назв

р := p+1

пл := страны_i.числ / страны_i.площ

р := 0

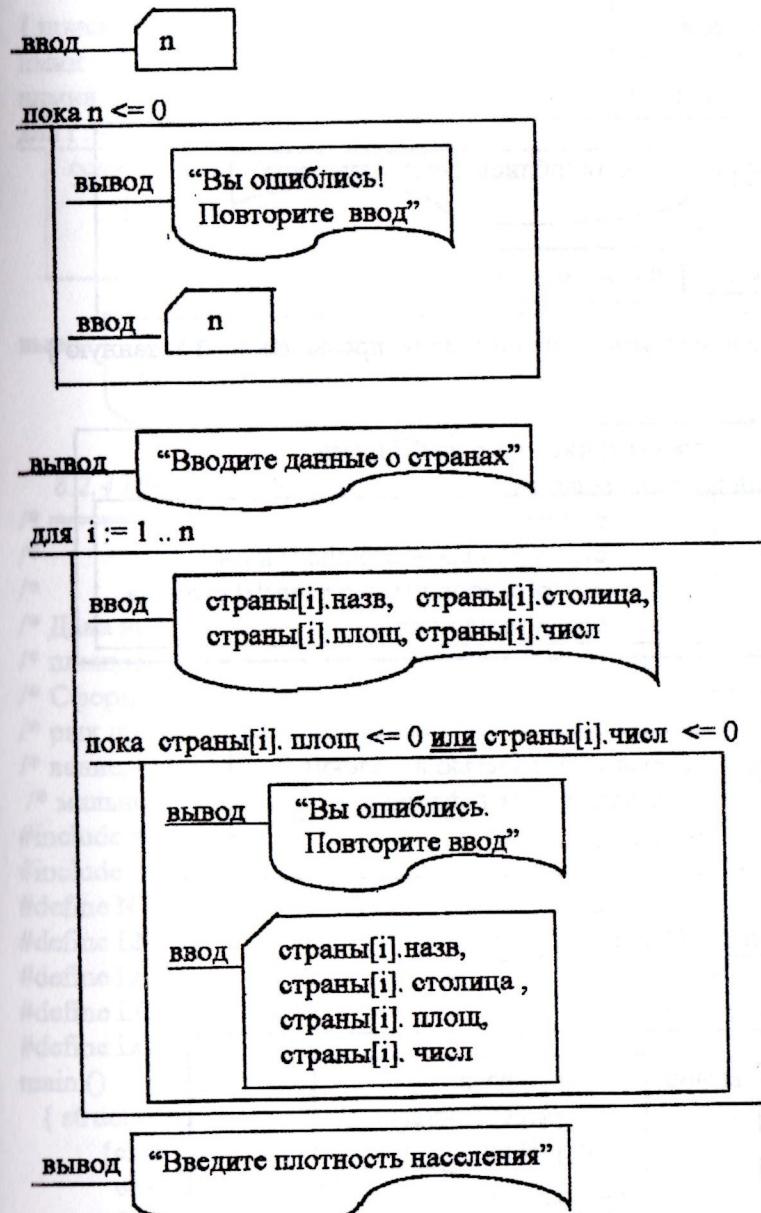
пл > пл_задан | i = 1..n

Промежуточные данные:

i: целое;	{ индекс элемента таблицы }
пл: вещественное;	{ плотность населения для i-й страны }
шлмин: вещественное;	{ значение текущего минимума }
нмин: целое;	{ номер элемента таблицы, соответствующего текущему минимуму }

6.2.3 Алгоритм

вывод "Введите количество стран"



ввод пл_задан

пока пл_задан <= 0

вывод "Вы ошиблись. Повторите ввод"

ввод пл_задан

{ поиск стран с плотностью населения, превышающей заданную }

р := 0

для i := 1 .. n

пл := страны[i].числ / страны[i].площ
если пл > пл_задан то
 р := р+1
 стр_p[r].назв := страны[i].назв
 стр_p[r].столица := страны[i].столица
 стр_p[r].пл := пл

если р = 0 то

вывод "Стран с плотностью населения",
пл_задан, "не найдено"

иначе

вывод "Список стран"

для i := 1 .. р

вывод стр_p[i].назв,
стр_p[i].столица,
стр_p[i].пл

{ поиск страны с минимальной площадью территории }

нмин := 0

плмин := 1000000

для i := 1 .. n

если стр[i].площ < плмин

нмин := i

плмин := стр[i].площ

вывод "Страна с минимальной площадью населения",
стр_p[nмин].назв

6.2.4 Программа на языке Си

```
/* ===== РАБОТА С ТАБЛИЧНЫМИ ДАННЫМИ ===== */
/* ЛАБОРАТОРНАЯ РАБОТА N 5 */
/*
    студентки гр. МИ93 Ивановой С.
*/
/* Даны информация о странах мира в виде: название, столица,
 * площадь (млн. кв. км) и численность населения (млн. чел.). */
/* Сформировать список тех стран, плотность населения которых
 * превышает заданную. Об этих странах сообщить их название,
 * столицу и плотность населения. Найти страну с минимальной
 * площадью территории. */

#include <stdio.h>
#include <stdlib.h>
#define N 100
#define LN 20
#define LC 15
#define LG_I 66
#define LG_V 51
main ()
{
    struct country /* структурный тип для исходной таблицы: */
    {
        char name [LN+1];           /* название */
        char capit [LC+1];          /* столица */
        float area;                 /* площадь */
        float chisl;                /* численность */
    };
    /* ... */
}
```

```

/* ===== ИСХОДНЫЕ ДАННЫЕ ===== */
int n; /* реальное количество стран */
struct country c[N]; /* исходная таблица */
float pl_z; /* заданная плотность населения */
/* ===== РЕЗУЛЬТАТЫ ===== */
struct
{
    char name [LN+1]; /* название */
    char capit [LC+1]; /* столица */
    float pl; /* плотность населения */
}c_r[N]; /* таблица-результат */
int p; /* реальное количество стран в таблице-результате */
/* ===== ПРОМЕЖУТОЧНЫЕ ДАННЫЕ ===== */
int i; /* индекс элемента таблицы */
float pl; /* плотность населения для i-й страны */
float ar_min; /* значение текущего минимума */
int n_min; /* номер элемента таблицы, соответствующего текущему минимуму */
int pr; /* признак корректности введенных данных */
float r; /* переменная для ввода данных с плавающей точкой */
/* отчеркивание при выводе исходной таблицы: */
char sg_icx[LG_I+1]=
"-----";
/* отчеркивание при выводе таблицы-результата: */
char sg_vix[LG_V+1]=
"-----";
/* ===== ПРОВЕРКА ОГРАНИЧЕНИЙ ===== */
/* ----- для количества стран в таблице ----- */
do
{
    printf("Введите количество стран (не более %d):", N);
    scanf("%d", &n);
    if (n<0 || n>N)
        {pr=1; printf("Вы ошиблись!\n");}
    }
else pr=0;
}
while(pr);

```

```

/* ----- для элементов таблицы данных ----- */
printf("Введите данные о странах\n");
printf("%s\n", sg_icx);
printf("INN! название страны ! столица !");
printf("площадь ! численность !\n");
printf("%s\n", sg_icx);
do
{
    for(i=0; i<n; i++)
    {
        printf("! %d ", i+1);
        scanf("%s", c[i].name);
        scanf("%s", c[i].capit);
        scanf("%f", &r); c[i].area=r;
        scanf("%f", &r); c[i].chisl=r;
        if(c[i].area<0 || c[i].chisl<0)
            {pr=1;
            printf("Вы ошиблись! Повторите ввод\n");
            i--;
            }
        else pr=0;
    }
}
while(pr);
printf("%s\n", sg_icx);
/* ----- для заданной плотности населения ----- */
do
{
    printf("Введите плотность населения:");
    scanf("%f", &pl_z);
    if(pl_z<0)
        {pr=1; printf("Вы ошиблись! Повторите ввод\n");}
    }
else pr=0;
}
while(pr);
/* ===== ИСПОЛНИМАЯ ЧАСТЬ АЛГОРИТМА ===== */
n_min = -1;
ar_min = 1.0e6; /* площадь, превышающая все известные */

```

```

p = -1;
for(i=0; i<n; i++)
{ /* Страны с плотностью, превышающей заданную */
    pl=c[i].chisl/c[i].area;
    if(pl>pl_z)
    {
        p++;
        strcpy(c_r[p].name,c[i].name);
        strcpy(c_r[p].capit,c[i].capit);
        c_r[p].pl=pl;
    }
/* ===== Поиск страны с минимальной площадью ===*/
    if(c[i].area<ar_min)
        {ar_min=c[i].area;
        n_min=i;
        }
}
/* ===== ВЫВОД РЕЗУЛЬТАТОВ=====*/
/* Страны с плотностью населения, превышающей заданную */
if(p>-1)
{
    printf("Список стран с площадью населения,"
           " превышающей %f\n", pl_z);
    printf("%s\n", sg_vix);
    printf("!NN! название страны ! столица ! плотность!\n");
    printf("%s\n", sg_vix);
    for(i=0;i<p;i++)
        printf("%2d%20s%-15s% 7.2f !\n",
               i+1, c_r[i].name, c_r[i].capit, c_r[i].pl);
    printf("%s\n", sg_vix);
}
else
    printf("Стран с плотностью населения, превышающей %f"
           " не найдено.\n", pl_z);
/* ===== Страна с минимальной площадью ===*/
printf("Страна с минимальной площадью населения: %s",
       c[n_min].name);
}

```

6.3 Варианты заданий

Во всех вариантах после текста собственно задания в скобках перечислены названия столбцов таблицы-результата.

1. Показатели развития торговли в СССР. - МСЭ, т. 9, с. 411.

Показатель	1932	1940	1950	1958
Объем розничного товарооборота	40.4	175.1	359.6	677
Число предприятий (тыс.)	340.2	494	511	650
Численность работников (тыс.)	1488	2166	1967	2847
Издержки обращения (%)	12.9	9.7	7.7	6.6

а) Абсолютный прирост какого показателя развития торговли за период 1940-1950гг. имел максимальное значение? [показатель, его значения в 1940г. и 1950г., прирост].

б) Какие показатели развития торговли СССР не превысили значения P единиц в 1932г.? [показатель, его значение в 1932г., единица измерения].

2. Здравоохранение в СССР (тыс.). - МСЭ, т. 7, с. 913.

Республика	1940		1960	
	Число врачей	Число коек	Число врачей	Число коек
РСФСР	82.2	482	221.2	939.4
Украина	33.4	157.6	73.8	318
Латвия	1.7	12	5.2	23.2
Эстония	0.07	0.1	2.7	11.2

а) В какой республике СССР значение отношения числа врачей к числу коек в 1940г. было минимальным? [республика, число врачей, число коек, отношение числа врачей к числу коек (все данные за 1940г.)].

б) В каких республиках СССР число врачей в 1960г. было меньше M тыс.? [республика, число врачей].

3. Производство сельскохозяйственных продуктов. - МСЭ, т. 7, с. 359.

Продукция (млн.т.)	Южная Америка		Австралия	
	1953	1957	1953	1957
Виноград	2.6	2.7	0.5	0.5
Мясо и сало	5.4	6.3	1.9	2.1
Молоко	13.1	15	11.1	11.5
Шерсть	0.3	0.3	0.8	0.9
Яйца (млрд. шт.)	9.1	11.1	3	3

а) Производство какого сельскохозяйственного продукта в 1953г. в Южной Америке было максимальным? [продукт, объем производства в 1953 г., единица измерения].

б) Относительный прирост производства каких сельскохозяйственных продуктов за период 1953-1957гг. в Австралии превысил X ? [сельскохозяйственный продукт, объем производства в 1953г. и 1957г., прирост].

4. Добыча полезных ископаемых в США. - МСЭ, т. 7, с. 706.

Ископаемые	Единица измерения	1937	1943	1957
Каменный уголь	млн. тонн	451	503	458
Нефть	млн. тонн	173	293	354
Газ	млрд. м ³	70	99.5	300
Железная руда	млн. тонн	73	103	106
Марганец	тыс. тонн	41	186	332
Медь	тыс. тонн	764	990	992
Свинец	тыс. тонн	422	44	306
Цинк	тыс. тонн	568	675	483

а) Относительный спад добычи какого полезного ископаемого в США за период 1943-1957гг. был максимальным? [наименование полезного ископаемого, объем добычи в 1943г. и 1957 г., единица измерения, спад добычи].

б) Добыча каких полезных ископаемых в США в 1937г. не превысила Y млн. тонн? [наименование полезного ископаемого, объем добычи в 1937г., единица измерения].

5. Посевные площади зерновых культур СССР (млн. га). - МСЭ, т. 3, с.123.

Культуры	1913	1928	1940	1950	1955
Рожь	28.2	24.1	23.1	23.6	19.1
Пшеница	23	27.7	40.3	38.5	60.5
Кукуруза	2.2	4.4	3.6	4.8	9.1
Ячмень	13.2	7.3	11.3	8.6	9.9
Овес	19.1	17.2	20.2	16.2	14.8
Гречиха	2.2	2.9	2	3	2.8
Прясе	3.5	5.7	6	3.8	7.7

а) Абсолютный прирост посевных площадей какой зерновой культуры за период 1913-1928гг. был минимальным? [зерновая культура, ее посевная площадь в 1913г. и 1928г., прирост].

б) Посевная площадь каких зерновых культур в СССР в 1955г. превысила Z млн. га ? [зерновая культура, ее посевная площадь в 1955 г.].

6. Производство продукции тяжелой промышленности в СССР. - МСЭ, т. 9, с. 637.

Вид	Единицы измерения	1913	1928	1940	1958
Чугун	млн. тонн	4.2	3.3	14.9	39.6
Сталь	млн. тонн	4.2	4.3	18.3	54.9
Уголь	млн. тонн	29.1	35.5	165	496
Нефть	млн. тонн	9.2	11.6	31.1	113
Газ	млрд. м ³	0.02	0.33	3.39	29.9

а) Абсолютный прирост производства какого вида продукции тяжелой промышленности за период 1913-1928гг. был максимальным? [продукция, производство в 1913г. и 1928г., единица измерения, прирост].

б) Производство каких видов продукции в 1958 г. превысило X млн. т.? [продукция, объем в 1958г., единицы измерения].

7. Производство продукции животноводства в СССР. - МСЭ, т. 3, с. 902.

Продукты	Единица измерения	1913	1940	1950	1958
Мясо и сало	млн. тонн	5	4.7	4.9	7.9
Молоко	млн. тонн	29.4	33.6	35.3	57.8
Шерсть	тыс. тонн	192	161	180	321
Яйца	млрд. штук	11.9	12.2	11.7	23.5

а) Относительный прирост производства какого продукта животноводства за период 1913-1950гг. был минимальным? [наименование, объем производства в 1913г. и 1950г., прирост].

б) Производство каких продуктов животноводства в 1958г. превысило P млн. тонн? [наименование продукта животноводства, объем его производства в 1958г., единица измерения].

8. Темпы роста производительности труда рабочих в промышленности (1913г.=100%). - МСЭ, т. 10, с. 762.

Годы	СССР	США	Англия	Франция
1928	120	137	94	105
1940	422	166	105	114
1950	580	209	122	131
1955	837	250	139	172

а) В каком году разница в темпах роста производительности труда во Франции и Англии максимальна? [год, рост, разница].

б) В каких годах темп роста производительности труда в США не превысил $Y\%$? [год, процент роста].

9. Добыча минерального топлива в СССР. - МСЭ, т. 9, с. 402.

Годы	Уголь (млн. т)	Нефть (млн. т)	Газ (млрд. м ³)	Сланцы (тыс. т)
1913	29.1	9.2	-	-
1928	35.5	11.6	0.3	0.6
1940	165	31.1	3.2	1662.9
1950	261	37	5.6	4710

а) В каком году в СССР добыли меньше всего угля? [год, объем добычи угля].

б) В каких годах в СССР добыли не менее N млн. т. нефти? [год, объем добычи нефти].

10. Удельный вес стран в мировом капиталистическом экспорте (% к итогу). - МСЭ, т. 9, с. 412.

Страны	1883	1913	1929	1937	1946	1956
США	10.7	13.3	15.6	12.9	30.1	20.7
Англия	15.6	13.9	10.8	10	11.6	9.7
Франция	8.9	7.2	6	3.7	2.7	4.9
Германия	10.3	13.1	9.7	9.3	0.7	8.0
Япония	0.5	1.7	2.9	4.7	0.3	2.7

а) Удельный вес какой страны в мировом экспорте в 1956 г. был максимальным? [страна, ее удельный вес в 1956 г.].

б) Относительный прирост удельного веса каких стран в мировом капиталистическом экспорте за период 1883-1913 гг. не превысил $Y\%$? [страна, удельный вес в 1883 г. и 1913 г., прирост].

11. Поголовье продуктивного скота (на 100 га угодий) в СССР. - МСЭ, т. 3, с. 901.

Виды	1940	1950	1955
Крупный рогатый скот, из них коровы	11	12	14
	6	5	6
Свиньи	15	13	25
Овцы	17	17	26
Козы	2	4	3

а) Поголовье какого вида продуктивного скота в СССР в 1955 г. было минимальным? [вид продуктивного скота, его поголовье в 1955 г.].

б) Абсолютный спад поголовья каких видов продуктивного скота в СССР за период 1940-1950 гг. не превысил S^* ? [вид скота, его поголовье в 1940 г. и 1950 г., величина спада].

12. Основные показатели развития промышленности Украины. - МСЭ, т. 9, с. 714.

Вид продукции	Единицы измерения	1913	1928	1940	1959
Чугун	млн. тонн	2.9	2.4	9.6	22.3
Сталь	млн. тонн	2.4	2.4	8.9	24
Прокат	млн. тонн	2.1	2	6.5	19.6
Железная руда	млн. тонн	6.9	4.7	20.2	53.5
Кокс	млн. тонн	4.4	4	15.7	29.2
Уголь	млн. тонн	22.8	24.8	83.8	167.7
Нефть	тыс. тонн	1047	-	353	1600
Газ	млрд. м ³	-	-	0.5	11.6

а) Производство какого вида продукции промышленности Украины в 1928 г. было максимальным? [продукция, объем ее производства в 1928 г., единица измерения].

б) Абсолютный прирост производства каких видов продукции Украины за 1940-59 гг. не превысил P^* ? [продукция, объем ее производства в 1940 г. и 1959 г., единицы измерения, прирост].

13. Государственный бюджет США (млрд. долларов). - МСЭ, т. 9, с. 1010.

Год	Всего доходов	Налоговые доходы	Всего расходов	Военные расходы
1938/39	5	4.8	8.9	1.1
1949/50	36.5	35.1	29.6	13
1955/56	68.2	65.2	66.5	40.6
1956/57	71	68.3	69.4	43.3
1957/58	69.1	65.9	71.9	44.1

а) В каком году в США собрали меньше всего налогов? [год, налоговые доходы].

б) В каких годах доля военных расходов бюджета превышала D млрд. долл.? [год, военные расходы, доля в расходной части].

14. Структура добычи минерального топлива (в % в пересчете на условное топливо). - МСЭ, т. 9, с. 404.

Вид топлива	СССР 1955	СССР 1956	США 1954
Каменные и бурые угли	68.6	62.1	32.7
Нефть	27.8	27.4	38.7
Природный газ	2.5	5.3	28.6
Торф	4.5	4.5	-
Горючие сланцы	0.6	0.7	-

а) Относительный прирост добычи какого топлива в СССР в 1955-1956гг. был максимальным? [вид топлива, добыча в 1955г. и 1956г., прирост].

б) Добыча каких видов минерального топлива в США была не меньше Y ? [вид топлива, объем его добычи в 1954 г.].

15. Международная торговля. - МСЭ, т. 9, с. 412.

Год	Оборот (млрд. \$)	Индекс экспорта (1953 год = 100)		
		США	Англия	ФРГ
1953	149	100	100	100
1954	154	93	101	123
1955	170	103	113	152
1956	188	112	112	171
1957	204	114	116	196
1958	193	119	116	205

а) В каком году оборот был минимальным? [год, оборот].

б) В каких годах разница в индексах экспорта США и ФРГ была не больше P ? [год, индекс экспорта США и ФРГ, разница].

16. Производство цемента (млн. т). - МСЭ, т. 10, с. 218.

Страна	1937	1958
Китай	2.3	9.3
Польша	1.3	5.0
Чехословакия	1.3	4.1
Венгрия	0.3	1.3
Болгария	0.2	0.9
СССР	5.7	33.3

а) В какой стране в 1958г. произвели больше всего цемента? [страна, объем производства цемента в 1958г.].

б) В каких странах абсолютный прирост производства цемента за период 1937-1958гг. был не меньше Z ? [страна, объем производства в 1937г. и 1958г., прирост].

17. Производство чугуна и стали (тыс. тонн). - МСЭ, т. 10, с. 2.

Страны	Чугун		Сталь	
	1938	1958	1938	1958
Китай	945	13690	488	11080
Чехословакия	1675	3774	2118	5510
Польша	880	3864	1440	5631
Венгрия	335	1082	647	1627
Румыния	132	737	284	934

а) В какой стране относительный прирост производства чугуна был минимальным? [страна, объем производства чугуна в 1938г. и 1958г., прирост].

б) В каких странах производство стали в 1938г. не превысило S тыс. т.? [страна, объем производства стали в 1938г.].

18. Урожайность сельскохозяйственных культур в СССР (ц/га). - МСЭ, т. 7, с. 899.

Культуры	1913	1928	1940	1945	1953
Зерновые	8.1	7.9	8.6	7.9	7.8
Сахарная свекла	168	132	146	159	148
Хлопок- сырец	10.8	8.1	10.8	15.3	20.5
Льно - волокно	3.3	2.4	1.7	1.3	1.3
Овощи	91	132	97	72	87

а) Абсолютный спад урожайности какой культуры за период 1940-1945гг. был максимальным? [культура, урожайность в 1940г. и 1945г., величина спада].

б) Урожайность каких культур в 1913г. была меньше Y ц/га? [сельскохозяйственная культура, ее урожайность в 1913г.].

19. Производство электроэнергии и мощность электростанций СССР. - МСЭ, т. 10, с. 790.

Год	Производство (млн. кВт/ч)		Мощность (тыс. кВт)	
	всего	в т.ч. ГЭС	всего	в т.ч. ГЭС
1913	1945	35	1098	16
1928	5007	420	1905	121
1940	48309	5113	11192	1587
1945	43257	4841	11124	1252
1955	170225	23165	37243	5996

а) В каком году производство всей электроэнергии СССР было минимальным? [год, производство электроэнергии (всего)].

б) В каких годах отношение мощности ГЭС к объему производства превысила M^7 [год, мощность и производство ГЭС].

20. Государственный бюджет Англии. - МСЭ, т. 9, с. 1010.

Год (млн. ф. ст.).	Расход всего	% военных расходов	Дефицит(-) или превышение (+) доходов	Государ- ственныи долг
1938/39	1106	22	-163	7289
1949/50	3928	18,9	+41	25986
1955/56	5253	26,8	-141	27520
1956/57	5704	26,8	-321	27280
1958/59	5988	24,5	-182	27300

а) В каком году дефицит бюджета Англии был максимальным? [год, величина дефицита].

б) В каких годах государственный долг Англии превышал M млн. ф. ст.? [год, государственный долг].

21. Посевная площадь и сбор основных культур в Японии. - МСЭ, т.10, с.1196.

Культура	Площадь, га		Сбор, тыс. тонн	
	1940	1950	1940	1950
Рис	3172	3011	9356	9652
Ячмень	778,3	1128	1617	1958
Картофель	143	192	1494	2407
Батат	243	399	2988	6292
Чай	39	27	48,4	36

а) Посевная площадь какой культуры в Японии в 1940г. была минимальной? [культура, площадь].

б) Урожайность каких культур в Японии в 1950г. не превышала Y тыс. т.? [культура, площадь, сбор, урожайность].

22. Энергетические мощности сельского хозяйства СССР (млн. л. сил). - МСЭ, т. 10, с. 934.

Вид	1940	1950	1958
Тракторы	17,6	22	42,6
Моторы комбайнов	5,8	8	22,2
Автомобили	11,9	21,3	58
Электроустановки	1,6	3,4	9,7

а) Мощность какого вида сельскохозяйственной техники СССР в 1940г. была максимальной? [вид, его мощность в 1940г.].

б) Относительный прирост мощности каких видов техники за период 1950-1958гг. превышал X ? [вид техники, его мощность в 1950г. и 1958г., прирост].

23. Основные показатели развития промышленности Украины. - МСЭ, т. 9, с. 714.

Вид продукции	Единицы измерения	1913	1928	1940	1945
Электроэнергия	млрд. кВт/ч	0.5	1.3	12.4	3.1
Сода кальцинированная	тыс. тонн	119	175.7	434	128.7
Минеральные удобрения	тыс. тонн	36	57	1012	136
Комбайны угольные	шт.	-	-	22	-
Тракторы	тыс. шт.	269	297	1218	335

а) Абсолютный прирост производства какой продукции промышленности Украины за период 1913-1928гг. был минимальным? [продукция, производство в 1913г. и 1928г., единица измерения, прирост].

б) Производство каких видов продукции промышленности Украины в 1945г. было не больше Y тыс. т.? [вид продукции, объем ее производства в 1945г., единицы измерения].

24. Производство продукции тяжелой промышленности в СССР. - МСЭ, т. 9, с. 637.

Вид продукции	Единица измерения	1913	1928	1940	1958
Станки	тыс. шт.	1.5	2	58.4	138
Турбины	тыс. кВт	5.9	44.1	1179	6031
Экскаваторы	шт.	-	-	274	10105
Цемент	млн. тонн	1.5	1.8	5.7	33.1
Автомобили	тыс. шт.	-	0.84	145.4	511
Тракторы	тыс. шт.	-	1.3	31.6	219.7

а) Производство какой продукции тяжелой промышленности в 1928г. было максимальным? [вид продукции, объем ее производства в 1928г., единица измерения].

б) Относительный прирост производства каких видов продукции за период 1940-1958гг. был меньше Z ? [вид продукции, ее производство в 1940г. и 1958г., единицы измерения, прирост].

25. Производство продукции электротехнической промышленности СССР. - МСЭ, т. 10, с. 878.

Вид продукции	Единицы измерения	1928	1940	1945	1958
Генераторы и турбины	тыс. кВт	75	468	265	5186
Электродвигатели мощностью выше 100 кВт	тыс. шт.	0.4	3.1	3.2	16.3
Электродвигатели мощностью до 100 кВт	тыс. шт.	32.8	259.3	110.7	2215
Холодильники бытовые	тыс. шт.	-	3.5	0.3	359

а) Абсолютный прирост производства какого вида продукции за период 1928-1940г. был минимальным? [вид продукции, объем ее производства в 1928г. и 1940г., единица измерения, прирост].

б) Производство каких видов продукции электротехнической промышленности в 1945г. было больше K тыс. штук? [вид продукции, объем ее производства в 1945г., единицы измерения].

26. Бюджет СССР (млрд. руб.) - МСЭ, т. 9, с. 1011.

Год	Доход	Расход	Расход на оборону
1940	180.2	174.4	56.8
1945	302	298.6	128.2
1950	422.8	413.2	82.8
1958	643	627.8	96.3

а) В каком году доля расходов на оборону в расходной части бюджета была максимальной? [год, расходы на оборону, доля].

б) В каких годах доход бюджета СССР превысил X млрд. рублей? [год, доход бюджета].

27. Уровень химического производства в 1958 г. (тыс. т.). - МСЭ, т. 10, с. 50.

Вид продукции	США	ФРГ	Италия
Минеральные удобрения	26872	12446	4308
Серная кислота	14442	2918	1950
Кальцинированная сода	4495	902	472
Пластмассы	2113	643.8	165
Искусственное волокно	682	238	156

а) Производство какого вида химической продукции в ФРГ было минимальным? [вид продукции, объем ее производства].

б) Разница в производстве каких видов продукции в США и Италии не превысила P тыс.т.? [продукция, объемы ее производства в США и Италии, разница].

28. Производство трикотажа в СССР. - МСЭ, т. 9, с. 502.

Вид продукции	1928	1940	1950	1958
Чулочно-носочные (млн. пар)	67.7	48.54	472.7	887
Бельевой трикотаж (млн. шт.)	6.9	124.4	150.4	399
Верхний трикотаж (млн. шт.)	1.4	58.6	47.1	97.2

а) Относительный прирост производства какого вида трикотажной продукции в СССР за период 1950-1958г. был максимальным? [вид продукции, объем ее производства в 1950г. и 1958г., единица измерения, прирост].

б) Производство каких видов трикотажной продукции в СССР в 1928г. было меньше X млн. штук? [вид продукции, объем ее производства в 1928г.].

29. Внешняя торговля Чехословакии. - МСЭ, т. 10, с. 415.

Показатель (в % от стоимости)	1948		1958	
	Экспорт	Импорт	Экспорт	Импорт
Машины и оборудование	20.3	7.2	43.4	18.7
Топливо и сырье	43.5	56.5	31.1	54.7
Продовольственные товары	5.5	33.6	7.1	23.1
Остальные товары	30.2	2.7	18.4	3.5

а) Экспорт какого товара Чехословакии в 1958 г. был минимальным? [наименование товара, его экспорт в 1958г.].

б) Отношение импорта к экспорту каких видов товаров в 1948г. превысило X ? [наименование, показатели, их отношение].

30. Посевные площади в СССР (млн. га). - МСЭ, т. 7, с. 899.

Культуры	1913	1928	1940	1945	1953
Зерновые	94.3	92.2	110.5	83.3	106.7
Картофель	3.1	5.7	7.7	8.1	8.3
Овощи	0.5	0.8	1.3	1.8	1.3
Кормовые	2.1	3.9	18.1	10.2	28.7

a) Абсолютный прирост посевной площади какой культуры за период 1928-1940гг. был максимальным? [культура, ее посевная площадь в 1928г. и 1940г., прирост].

б) Посевные площади каких культур в 1953 г. превысили P млн. га? [наименование культуры, ее посевная площадь в 1953г.].

31. Хлопковое производство в СССР. - МСЭ, т. 10, с. 84.

Показатель	1913	1928	1940	1958
Посевная площадь (млн. га)	0.69	0.97	2.08	2.15
Заготовки и закупки (млн. т)	0.68	1.03	2.51	4.4
Сбор с 1 га (т)	13	6.8	12.1	20.4

а) За какой период абсолютный прирост сбора хлопка с 1 га в СССР был минимальным? [период, сбор с 1 га].

б) В каких годах посевная площадь хлопка в СССР не превысила P млн. га? [год, посевная площадь].

32. Государственный бюджет и долг Франции (млрд. фр.) - МСЭ, т. 9, с. 1010.

Год	Расход	Дефицит	Государственный долг
1938	82	-28	424
1950	2357	-280	4133
1955	3945	-495	5887
1956	4643	-765	5495
1957	5640	-655	7188
1958	5486	-258	8144

а) В каком году расходная часть бюджета Франции была максимальной? [год, величина расхода].

б) В каких годах разница между долгом и дефицитом бюджета не превышала S млрд. фр.? [год, долг, дефицит, разница].

33. Основные показатели развития промышленности Украины. - МСЭ, т. 9, с. 714.

Вид продукции	Единицы измерения	1913	1928	1940	1959
Автомобили грузовые	тыс. шт.	-	-	-	12.6
Цемент	тыс. тонн	269	297	1218	7017
Кирпич	млрд. шт.	0.6	0.7	1.6	6.3
Ткани х/б	млн. м	4.7	2	13.8	88.2
Ткани шерстяные	млн. м	5.3	2	12	17.7
Сахар-песок	тыс. тонн	1107	1041	1580	4103

а) Производство какого вида продукции в Украине в 1913г. было минимальным? [вид продукции, объем ее производства в 1913г., единица измерения].

б) Относительный прирост производства каких видов продукции за период 1940-1959гг. был больше X ? [вид продукции, ее производство в 1940г. и 1959г., единица измерения, прирост].

34. Производство электроэнергии - МСЭ, т. 10, с. 793.

Страны (млрд. кВт/ч.).	1937	1950	1955
Китай	6	4.3	12.3
Польша	3.6	9.4	17.8
Чехословакия	4.1	9.3	15
Венгрия	1.4	3	5.4
Румыния	1.1	2.1	4.3
Болгария	0.3	0.8	2.1

а) В какой стране абсолютный прирост производства электроэнергии за период 1950-1955г. был максимальным? [страна, объем производства в 1950г. и 1955г., прирост].

б) В каких странах производство электроэнергии в 1937г. не превысило Z млрд. кВт/ч.? [страна, объем производства в 1937г.].

35. Выпуск промышленной продукции в Японии. - МСЭ, т.10, с. 1196.

Вид продукции	Единицы измерения	1937	1950	1958
Медь	тыс. тонн	86.7	84.7	123.7
Цинк	тыс. тонн	49.2	49	143
Свинец	тыс. тонн	10.3	16	41.3
Алюминий	тыс. тонн	10.7	28.3	84.5
Серная кислота	млн. тонн	3.5	3.2	3.8
Цемент	млн. тонн	6.1	4.4	14.98

а) Абсолютный прирост производства какого вида продукции в Японии за период 1950-1958гг. был минимальным? [вид продукции, объем ее производства в 1950г. и 1958г., единица измерения, прирост].

б) Производство каких видов продукции в Японии в 1937г. было больше Z единиц? [вид продукции, объем ее производства в 1937г., единица измерения].

36. Этнический состав населения городов СССР в - [3], с.66.

Город (1970 г.)	Число жителей (тыс.)	% коренной националь- ности	% русских	% других национальностей
Киев	1632	64.8	22.9	12.3
Минск	917	65.6	23.4	11.0
Баку	1256	46.3	27.7	26.0
Тбилиси	889	57.5	14.0	28.5
Ташкент	1385	37	40.8	22.2

а) В каком городе численность жителей коренной национальности в 1970г. была максимальной? [город, общее число жителей, численность жителей коренной национальности].

б) В каком городе разность доли жителей русской и других национальностей менее P процентов? [город, численность жителей русской национальности, численность жителей других национальностей].

37. Распределение городского населения СССР по городам разных размеров, % - [3], с.15.

Размер города, тыс.чел	1925 г.	1959 г.	1970 г.	1974 г.
До 50	48	40	35	33
50-100	16	11	10	10
100-500	20	25	28	29
500 и более	16	24	27	28

а) Города какого размера имеют минимальный абсолютный прирост доли городского населения за период 1959-1970гг.? [размер города, доля городского населения в 1959г. и 1970г.].

б) В каких городах доля городского населения в 1925г. составляла больше D %? [размер города, доля горожан в 1925г.].

38. Динамика числа разводов в СССР. - [3], с.60.

Показатель	1950 г.	1960 г.	1965 г.	1970 г.
Число браков, тыс.	2081	2592	2009	2365
Число разводов, тыс.	67	270	360	636
Браков на 1000 чел.	11.6	12.1	8.7	9.7
Разводов на 1000 чел.	0.4	1.3	1.6	2.6

а) За какой период абсолютный прирост числа браков был максимальным? [годы периода, прирост числа браков].

б) В каких годах отношение числа браков на 1000 чел. к числу разводов на 1000 чел. не превысило значения B ? [год, числа браков на 1000 чел., число разводов на 1000 чел.].

39. Доля городского населения у национальностей СССР, % - [3], с.65.

Национальность	1925 г.	1939 г.	1959 г.	1970 г.
Русские	21	38	58	68
Украинцы	11	29	39	48
Белорусы	10	21	32	44
Армяне	35	41	57	65
Якуты	2	7	17	21
Карелы	3	14	31	45

а) У какой национальности относительный прирост доли городского населения за период 1925-1970гг. был минимальным? [национальность, доля городского населения в 1925г. и 1970г.].

б) У каких национальностей доля городского населения в 1939г. была больше N %? [национальность, доля городского населения в 1939г.].

40. Число рождений на 1000 женщин в СССР в 1969-1970гг. - [3], с.56.

Возрастная группа, лет	Город	Село
До 20	28.5	33.8
20-24	144.2	209.5
25-29	108.8	163.5
30-34	68.6	121.9
35-39	29.6	75.5
40-44	7.3	27.0
45-49	1.1	5.5

а) В какой возрастной группе отношение рождаемости в городе к рождаемости в селе максимальное ? [возрастная группа, рождаемость в городе, рождаемость в селе].

б) В каких возрастных группах рождаемость в городе не превысила R человек ? [возрастная группа, рождаемость в городе].

6.4 Контрольные вопросы по лабораторной работе

- 1) Объявите на языке Си структурный тип данных country примера 6.2 с помощью инструкции *typedef*.
- 2) Организовать на языке Си вывод списка стран из примера 6.2, названия которых начинаются на заданную букву, с указанием для результата названия столицы и численности населения.
- 3) Сравните преимущества использования первого и второго способа выбора промежуточных переменных при решении задачи о нахождении элемента таблицы, соответствующего минимуму (максимуму) значений в некотором столбце.

6.5 Содержание отчета

1. Условие задачи.
2. Постановка задачи.
3. Метод решения задачи.
4. Алгоритм.
5. Макеты шапок для всех таблиц задачи.
6. Текст программы на языке Си.
7. Таблица трассировки программы.
8. Выводы.
9. Программный документ "Описание программы".

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Единая система программной документации . - М.: Государственный комитет СССР по стандартам, 1982. - 128 с.
2. Абрамов С.А., Гнездилова Г.Г., Капустина Е.Н. и др. Задачи по программированию. - М.: Наука, 1988. - 224с.
3. Переведенцев В.И. Города и время. - М.: Статистика, 1975. - 87с.
4. Банахан М., Раттер Э. Введение в операционную систему UNIX. - М.: Радио и связь, 1985. - 344с.
5. Баурн С. Операционная система UNIX. - М.: Мир, 1985. - 344с.
6. Берри Р., Миккинз Б. Язык Си: Введение для программистов. - М.: Финансы и статистика, 1988. - 191с.
7. Болски М.И. Язык программирования Си: Справочник. - М.: Радио и связь, 1988. - 96с.
8. Готье Р. Руководство по операционной системе UNIX. - М.: Финансы и статистика, 1985. - 232с.
9. Джехани Н. Программирование на языке Си. - М.: Радио и связь. - 270с.
10. Инструментальная мобильная операционная система ИНМОС/ М.И.Беляев и др - М.: Финансы и статистика, 1985. - 231с.
11. Керниган Б., Ритчи Д. Язык программирования Си. - М.: Финансы и статистика, 1990. - 230с.
12. Керниган Б., Ритчи Д., Фьюэр А. Язык программирования Си. Задачи по языку Си. - М.: Финансы и статистика, 1985. - 279с.
13. Кристиан К. Введение в операционную систему UNIX. - М.: Финансы и статистика, 1985. - 318с.
14. Погорелый С.Д., Слободянюк Т.Ф. Программное обеспечение микропроцессорных систем: Справочник. - Киев: Техника, 1989. - 301с
15. Свен Т. Освоение Borland C++ 4.5. Энциклопедия функций. - К.: Диалектика, 1996. - 320с.
16. Тихомиров В.П., Давидов М.И. Операционная система ДЕМОС. - М.: Финансы и статистика, 1988. - 208с.
17. Томас Р., Йейтс Дж. Операционная система UNIX: Руководство для пользователей. - М.: Радио и связь, 1986. - 352с.
18. Хэнкок Л., Кригер М. Введение в программирование на языке Си. - М.: Радио и связь, 1986. - 192с.
19. Уинер Р. Язык Турбо Си. - М.: Мир, 1991. - 384с.
20. Уэйт М., Прата С., Мартин Д. Язык Си: Руководство для начинающих. - М.: Мир, 1984. - 512с.

ПРИЛОЖЕНИЕ А

ГОСТ 19.402-78

ОПИСАНИЕ ПРОГРАММЫ

2. ... Структуру и оформление документа устанавливают в соответствии с ГОСТ 19.105-78.

Составление информационной части (аннотации и содержания) является обязательным.

3. Описание программы должно содержать следующие разделы:

- общие сведения;
- функциональное назначение;
- описание логической структуры;
- используемые технические средства;
- вызов и загрузка;
- входные данные;
- выходные данные.

В зависимости от особенностей документа допускается вводить дополнительные разделы или объединять отдельные разделы.

4. В разделе "Общие сведения" должны быть указаны:

- обозначение и наименование программы;
- программное обеспечение, необходимое для функционирования программы;
- языки программирования, на которых написана программа.

5. В разделе "Функциональное назначение" должны быть указаны классы решаемых задач и (или) назначение программы и сведения о функциональных ограничениях на применение.

6. В разделе "Описание логической структуры" должны быть указаны:

- алгоритм программы;
- используемые методы;
- структура программы с описанием функций составных частей и связи между ними;
- связи программы с другими программами.

Описание логической структуры программы выполняют с учетом текста программы на исходном языке

7. В разделе "Используемые технические средства" должны быть

указаны типы ЭВМ и устройств, которые используются при работе программы.

8. В разделе "Вызов и загрузка" должны быть указаны:

- способ вызова программы с соответствующего носителя данных;
- входные точки в программу.

Допускается указывать адреса загрузки, сведения об использовании оперативной памяти, объем программы.

9. В разделе "Входные данные" должны быть указаны:

- характер, организация и предварительная подготовка входных данных;
- формат, описание и способ кодирования входных данных.

10. В разделе "Выходные данные" должны быть указаны:

- характер и организация выходных данных;
- формат, описание и способ кодирования выходных данных.

11. Допускается содержание разделов иллюстрировать пояснительными примерами, таблицами, схемами, графиками.

...

Приложение Б
Образец титульного листа отчета по лабораторной работе

Министерство образования Украины
Донецкий Государственный Технический Университет
Кафедра прикладной математики и информатики

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №1
“Календарные задачи на языке Си”
по курсу “Технология программирования”

ВАРИАНТ 50

“ПОДСЧИТАТЬ КОЛИЧЕСТВО ДНЕЙ МЕЖДУ ДВУМЯ
ЗАДАННЫМИ ДАТАМИ”

Выполнил: студент группы МИ9б
Петров С.В.

(подпись) “__” 1997 г.

Принял: ассистент каф. ПМИ
Суворова И.П.
(подпись) “__” 1997 г.

1997

130

Приложение В
Пример реферата в отчете по лабораторной работе

РЕФЕРАТ

Отчет о лабораторной работе: 9 с., 2 рис., 1 табл., 2 приложения, 3 источника

Объект исследования - технологические этапы решения календарных задач на ПЭВМ.

Цель работы - освоить приемы алгоритмизации при организации разветвляющихся и повторяющихся вычислений (на примере решения календарных задач) при использовании целочисленной арифметики; приемы ввода исходных данных и проверки их ограничений; приемы инициализации исходных данных (скалярных и структурированных); приемы формирования и вывода результатов, количество которых формируется в программе динамически.

Метод исследования - метод перебора, метод накопления.

При использовании целочисленной арифметики на примере решения календарной задачи освоены:

- основные технологические этапы решения задач (постановка задачи; метод решения; алгоритм; кодирование алгоритма средствами языка Си; отладка и тестирование программы);

- приемы алгоритмизации при организации разветвляющихся и повторяющихся вычислений;

- приемы ввода исходных данных и проверки их ограничений;

- приемы инициализации скалярных и структурированных исходных данных.

Решена задача составления расписания консультаций перед экзаменами зимней сессии для студентов специальности 7.050207.

Результаты лабораторной работы внедрены на кафедре ПМИ при составлении расписания консультаций экзаменационной сессии студентов специальности “Информационные системы в менеджменте”.

**ТЕХНОЛОГИЧЕСКИЕ ЭТАПЫ, КАЛЕНДАРНЫЕ ЗАДАЧИ,
ЦЕЛОЧИСЛЕННАЯ АРИФМЕТИКА, ПЭВМ, ЯЗЫК СИ.**

СОДЕРЖАНИЕ

1. Основные положения	3
2. Лабораторная работа N 1. Календарные задачи на языке Си	9
3. Лабораторная работа N 2. Решение задач аналитической геометрии на языке Си	33
4. Лабораторная работа N 3. Работа с многочленами и матрицами на языке Си	51
5. Лабораторная работа N 4. Работа с символьными и текстовыми данными на языке Си	72
6. Лабораторная работа N 4. Работа с табличными данными на языке Си	96
Список рекомендуемой литературы	127
Приложение А. Описание программы. ГОСТ 19.402-78	128
Приложение Б. Образец титульного листа отчета по лабораторной работе	130
Приложение В. Пример реферата в отчете по лабораторной работе	131