Решение олимпиадных задач по информатике

Автор: Э.С. Ларина

0. Лекция: Предисловие

Волгоградские школы, за небольшим исключением, в качестве языка программирования, изучаемого в курсе информатики, ранее использовали Бейсик, теперь Паскаль. Изучение любого структурированного языка в школьном возрасте способствует формированию алгоритмического мышления, поэтому задача школьного учителя - не углубляться в тонкости и возможности конкретного языка программирования, а учить разрабатывать алгоритмы.

Вполне допустимо, что начинающий программист не помнит точного названия стандартных функций и процедур, всех типов данных и др. - для напоминания существует справочная система. В качестве такой справки ниже приведены некоторые важные для понимания материала, изложенного в данном курсе, описания. К ним можно обращаться при прохождении лекций, особенно при выполнении упражнений, данных в конце каждой лекции. Эти справочные материалы по языку программирования Паскаль предназначены для получения краткой информации по вопросам состава и синтаксиса базовых конструкций языка.

Типы данных

Целые числа			
Описатель типа	**Длина(байт)**	**Минимальное число**	**Максимальное число**
Integer	2 (знак)	-32768	+32767
Shortint	1 (знак)	-128	+127
Longint	4 (знак)	-2147483648	+2147483647
Byte	1 (без зн.)	0	255
Word	2 (без зн.)	0	65535

Строковые переменные			
Описатель типа	**Длина(байт)**	**Количество значений**	**Допустимые значения**
Char	1	256	литера (символ)

Логические переменные			
Описатель типа	**Длина(байт)**	**Количество значений**	**Допустимые значения**
Boolean	1	2	true, false

Вещественные переменные			
Описатель типа	**Длина(байт)**	**Число значащих цифр**	**Директива компилятора**
Real	6	11	не требуется
Single	4	7	{$N+}

Double	8	15	{$N+}
Extended	10	19	{$N+}
Comp	8	19 (цел.число,64-bit)	{$N+}

Арифметические операции

целочисленное деление			остаток от деления (модуль)			двоичный сдвиг влево			двоичный сдвиг вправо		
a	**B**	**a div b**	**a**	**b**	**a mod b**	**a**	**b**	**a shl b**	**a**	**b**	**a shr b**
10	20	0	10	20	10	10	2	40	10	1	5
40	**15**	**2**	40	15	10	32	1	64	32	2	8

Логические операции

операции булевой алгебры (высший приоритет)										
Not		**And**			**or**			**xor**		
A	**not a**	**a**	**b**	**a and b**	**a**	**b**	**a or b**	**a**	**b**	**a xor b**
False	True	false	false	False	false	false	false	false	false	false
true	False	false	true	False	false	true	true	false	true	true
		true	false	False	true	false	true	true	false	true
		true	true	true	true	true	true	true	true	false

операции отношения (низший приоритет)	
a = b	равно
A <> b	не равно
a < b	меньше
A <= b	меньше или равно (не больше)
a > b	больше
A >= b	больше или равно (не меньше)

Функции для обработки числовых переменных

Функция	Назначение	Пример вызова	Результат
Abs (число)	абс. значение числа	abs(-3.5)	+3.5
Arctan (тангенс-угла)	арктангенс числа	arctan(0)	0
Cos (угол)	косинус угла(рад.)	cos(pi)	-1
Exp (число)	Экспонента	exp(1)	2.718281828...
Frac (число)	дробная часть числа	frac(3.5)	0.5
Int (число)	целая часть числа	int(3.5)	3.0
Ln (число)	нат. Логарифм	ln(2.718281828)	1.0
Odd (число)	проверка нечетности	odd(3)	True
pi	число пи	pi	3.141592...
Random (число)	"случайное" число	random(10)	Число в [0;9]
Sin (угол)	синус угла(рад.)	sin(pi)	0

Sqr (число)	квадрат числа	sqr(2.0)	4.0
Sqrt (число)	квадратный корень	sqrt(25.0)	5.0

Функции и процедуры для обработки строковых переменных

Функция	Значение	Пример вызова	Результат
chr(номер-символа-n)	Символ номер n (#n)	chr(33)	'!'
ord (величина)	номер величины (код)	ord('!')	33
succ (величина)	Следующее значение в последовательности	succ('y')	'z'
pred (величина)	Предыдущее значение в последовательности	pred('y')	'x'
copy(s,p,n)	Выделить n символов из строки s начиная с позиции p	copy('роза',3,2)	'за'
concat(s1,s2,...sn)	Соединить строки (литеры) в одну строку (конкатенация)	concat('г','роза')	'гроза'
length (строка)	Длина строки [символ.]	length('роза')	4
pos(s1,s2)	номер позиции строки s1 внутри строки s2 (если не найдена, 0)	pos('за','роза')	3

Процедура	Назначение	Пример вызова	Результат
delete(s,p,n)	удалить n символов из строки s с позиции p	delete('роза',1,2)	'за'
insert(s1,s2,p)	вставить строку (литеру) s1 в строку s2 с позиции p	insert('г','роза',1)	'гроза'

Процедура (функция)	Назначение	Пример вызова	Результат
round (число)	округлить число	n := round(3.5)	4
trunc (число)	отсечь дробную часть	n := trunc(3.5)	3
str(n:p:q,s)	преобразовать число n в строку s	str(3.5:3,s)	s = '3.5'
val(s,n,p)	преобразовать строку s (литеру) в число n	val('+3.5',n,p)	n = 3.5; p = 0 p=место ошибки

Основы машинной графики

Пример: uses graph, crt;

Основные процедуры и функции			
Процедура (функция)	**Назначение**	**Пример вызова**	**Примечания**
d := detect	Определить тип графического режима (номер драйвера)	d := detect	d = драйвер экрана (bgi) (integer)
initgraph(d, m, путь-bgi)	Установить графический режим экрана	initgraph(d, m, 'c:\bgi')	m = режим экрана (vga) (integer)
cleardevice	Очистить экран и	cleardevice	

	отменить установки цвета	
setcolor (цвет)	Установить цвет линии (рисунка)	setcolor(magenta)
setbkcolor (цвет)	Установить цвет фона (очистки)	setbkcolor(0)
putpixel(x,y,цвет)	Точка (x,y)	putpixel(5, 5, red)
line(x1,y1,x2,y2)	Линия (x1,y1)- (x2,y2)	line(10,10,20,200)
lineto(x,y)	Чертить линию в (x,y)	lineto(100,200)
moveto(x,y)	Вести перо в (x,y)	moveto(nx, ny)
circle(x,y,радиус)	Окружность (x,y,r)	circle(x, y, 20)
arc(x,y,угл1,угл2,радиус)	Дуга окружности (x,y,r) от угла1 до угла2 (радиан)	arc(10,10,0,pi,5)
setfillstyle(s,цвет-заливки)	Установить стиль и цвет заливки	setfillstyle(1,green)
floodfill(x,y,цветграницы)	Залить область с границей (цвет) цветом заливки	floodfill(p,q, 10)
rectangle(x1,y1,x2,y2)	Прямоугольник (x1,y1)-(x2,y2)	rectangle(2,2,5,10)
bar(x1,y1,x2,y2)	Прямоугольник (x1,y1)-(x2,y2) с заливкой цвета	bar(2, 2, m, n)
closegraph	Закрыть графический режим экрана	closegraph

1. Лекция: Базовые формулы (зависимости) и задачи, решаемые с их помощью
Цель лекции: научиться применять некоторые формулы и зависимости (зависимость уменьшающегося значения переменной в теле цикла от увеличивающегося значения счетчика цикла, формулу для определения кратности двух чисел, формулу для нахождения длины отрезка по заданным координатам его концов) в решении классических задач.

Содержание

Очень часто в решениях задач необходимо использовать ту или иную зависимость, применить формулу. Не стоит сейчас рассматривать большой круг математических формул, которые когда-либо использовались в решении задач по программированию (их достаточно много). А вот на некоторых из них, так называемых "базовых", стоит остановиться:

- Зависимость уменьшающейся переменной X в теле цикла от увеличивающегося значения счетчика цикла
- Признак кратности
- Нахождение длины отрезка по заданным координатам его концов.

Многие из рассмотренных ниже задач, опирающиеся на эти зависимости и формулы являются классическими в информатике.

Зависимость уменьшающейся переменной X в теле цикла от увеличивающегося значения счетчика цикла i

Для установления зависимости уменьшающейся переменной x в теле цикла от счетчика цикла, проанализируем значения переменных на каждом шаге выполнения тела цикла (проиллюстрированные в табл. 1.1):

Таблица 1.1.

i	x	
1	5	$x = n (n = 5)$
2	4	$x = n - 1 (1 = i - 1)$
3	3	$x = n - 2 (2 = i - 1)$
4	2	$vx = n - 3 (3 = i - 1)$
5	1	$x = n - 4 (4 = i - 1)$

Итого $x = n - (i - 1)$

Фрагменты программ, в котором реализована эта зависимость:

Бейсик:	Паскаль:

…
for i=1 to n
x=n-i+1
next
…
	…
	for i:=1 to n do
	x:=n-i+1;
	…

Разбор задачи, приведенной ниже позволит закрепить полученные знания.

Задача "Палиндром": Определить, палиндром ли слово, введенное с клавиатуры (палиндром читается одинаково слева направо и справа налево).

Идея решения: Во введенной строке необходимо проверить - равны ли первый и последний символы, второй и предпоследний и т.д. (используя зависимость уменьшающейся переменной X в теле цикла от увеличивающегося значения счетчика цикла i).

Обратите внимание, что тело цикла выполняется n/2 раз (за один проход сравниваются 2 символа).

Программа на Бейсике:

```
input "введите слово"; a$
n=len (a$)
for i=1 to n/2
 if mid$ (a$,i,1)<> mid$ (a$,n-i+1,1) then k=1
next
if k=0 then print "палиндром" else print "не палиндром"
```

Программа на Паскале:

```
var a:string;
 k,n,i: integer;
begin
 writeln ('введите слово');
 readln (a);
 k:=0;
 n:=length(a);
 for i:=1 to (n div 2) do
  if copy(a,i,1) <> copy(a,n-i+1,1) then k:=1;
 if k=0 then writeln ('палиндром')
 else writeln ('не палиндром');
end.
```

Тест:

Дано:	Ротор
Результат	палиндром

Длина отрезка

Для нахождения длины отрезка, заданного координатами своих концов (см. рис. 1.1) воспользуемся теоремой Пифагора:

$$\sqrt{(x2-x1)^2+(y2-y1)^2}$$

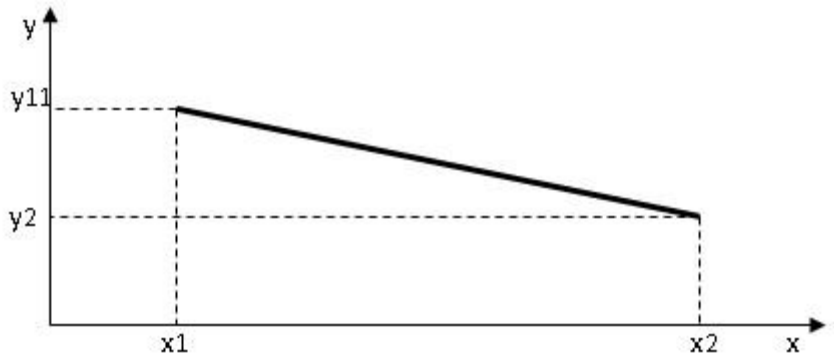

Рис. 1.1.

Арифметическое выражение для вычисления длины отрезка **на Бейсике**:

```
sqr ((x1-x2)^2+(y1-y2)^2)
```

Арифметическое выражения для вычисления длины отрезка **на Паскале**:

```
sqrt (sqr(x1-x2)+sqr(y1-y2))
```

Разбор решения задачи, приведенной ниже позволит закрепить полученные знания.

Задача: Найти периметр треугольника, координаты вершин которого вводятся с клавиатуры (рис. 1.2).

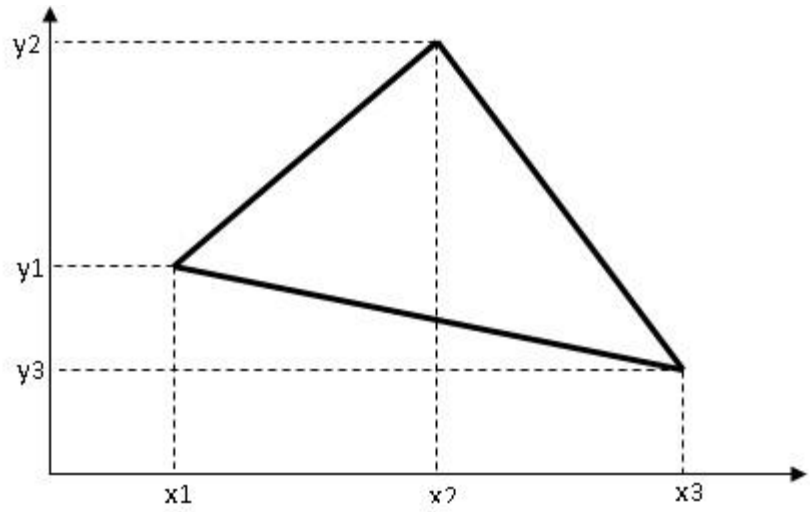

Рис. 1.2.

Идея решения: Для нахождения периметра треугольника необходимо найти длины его сторон.

Программа на Бейсике:

```
input x1, y1
input x2, y2
input x3, y3
ras1 = sqr((x1 - x2) ^ 2 + (y1 - y2) ^ 2)
ras2 = sqr((x2 - x3) ^ 2 + (y2 - y3) ^ 2)
ras3 = sqr((x1 - x3) ^ 2 + (y1 - y3) ^ 2)
print (ras1+ras2+ras3)
```

Программа на Паскале:

```
var   x1,y1,x2,y2,x3,y3: integer;
 ras1,ras2,ras3: real;
begin
 readln (x1, y1);
 readln (x2, y2);
 readln (x3, y3);
 ras1:=sqrt(sqr(x1-x2)+sqr(y1-y2));
 ras2:=sqrt(sqr(x2-x3)+sqr(y2-y3));
 ras3:=sqrt(sqr(x1-x3)+sqr(y1-y3));
 writeln (ras1+ras2+ras3);
end.
```

Тест:

Дано:	4,11
	3,7
	4,3
	7,10
	6,7
	6,2
Результат:	11,8
	9,5

Признак кратности числа X числу У

Логические выражения для определения кратности числа X числу У **на Бейсике:**

```
1. x \ y = x / y
2. x mod y =0
```

Логическое выражение для определения кратности числа X числу У **на Паскале:**

```
x mod y = 0
```

Разбор решения задачи, приведенной ниже позволит закрепить полученные знания.

Задача: Найти делители введенного с клавиатуры числа N.

Идея решения: Во всех приведенных ниже задачах (набора для практики) необходимо воспользоваться формулой n mod i=0 (проверить - равен ли остаток от деления нулю). Счетчик цикла "перебирает" возможные делители числа n.

Программа на Бейсике:

```
input "введите число"; n
print "делители:"
rem=вывод делителей========
for i=1 to n
 if n mod i=0 then print i; ",";
next
```

Программа на Паскале:

```
var n,i: integer;
begin
 writeln ('введите число');
 readln (n);
 writeln ('делители:');
 {=вывод делителей========}
for i:=1 to n do
  if n mod i=0 then write (i);
end.
```

Тест:

Дано:	16
Результат:	1 2 4 8 16

Задачи из раздела "Теория чисел"

Необходимые дополнительные сведения для решения задач:

- **Простое число** - число, большее чем 1 и делящееся только лишь на 1 и на само себя.
- **Совершенное число** - число, равное сумме своих делителей (не считать делителем числа само число).
- **Дружественные числа** - пара чисел, каждое из которых равно сумме правильных делителей другого.
- **Период строки** - повторяющаяся часть строки

Ниже приведены задачи из раздела "Теория чисел" с разбором решений.

Задача: Проверить - простое ли число, введенное с клавиатуры.

Программа на Бейсике:

```
input "введите число"; n
rem=нахождение количества  делителей============
for i=1 to n
 if n mod i=0 then k=k+1
next
if k=2 then print "число простое" else print "число составное"
```

Программа на Паскале:

```
program pr;
var n,i,k: integer;
begin
 writeln ('введите число');
 readln (n);
 k:=0;
 {=нахождение количества  делителей=}
 for i:=1 to Round (Sqrt(n)) do
 if n mod i=0 then k:=k+1;
 if k=2 then writeln ('число простое')
   else writeln ('число составное');
end.
```

Тест:

Дано:	8	11
Результат:	Число составное	Число простое

Задача: В диапазоне от A до B вывести все простые числа.

Программа на Бейсике:

```
input "введите границы диапазона "; a, b
print "простые числа:"
for n=a to b
 k=0
 for i=1 to n
  if n mod i=0 then k=k+1
 next i
 if k=2 then print n
next n
```

Программа на Паскале:

```
var a,b,n,i,k: integer;
begin
 writeln ('введите границы диапазона');
 readln (a,b);
 writeln ('простые числа:');
 for n:=a to b do
  begin
     k:=0;
     {=нахождение количества  делителей=}
     for i:=1 to Round (Sqrt(n)) do
         if n mod i=0 then k:=k+1;
     if k=2 then
     {
         writeln (n);
         writeln (' ');
     }
 end;
end.
```

Тест:

Дано:	10, 100

Результат:	11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97

Задача: Проверить - совершенное ли число, введенное с клавиатуры.

Программа на Бейсике:

```
input "введите число"; n
{========нахождение суммы  делителей================}
for i=1 to n/2
 if n mod i=0 then s=s+i
next
if s=n then print "число совершенное" else print "число несовершенное"
```

Программа на Паскале:

```
var n,i,s: integer;
begin
 writeln ('введите число');
 readln (n);
 s:=0;
 {==нахождение суммы  делителей==}
 for i:=1 to (n div 2) do
     if n mod i=0 then s:=s+i;
 if s=n then writeln ('число совершенное)
 else writeln ('число несовершенное);
end.
```

Тест:

Дано:	6
Результат:	Число совершенное

Задача: На диапазоне от A до B вывести все совершенные числа.

Программа на Бейсике:

```
input "введите границы диапазона "; a, b
print "простые числа:"
for n=a to b
 s=0
 rem=нахождение суммы  делителей=
 for i=1 to n/2
    if n mod i=0 then s=s+i
 next i
 if s=n then print n
next n
```

Программа на Паскале:

```
var a,b,n,i,s: integer;
begin
 writeln ('введите границы диапазона');
 readln (a,b);
 writeln ('совершенные числа:');
 for n:=a to b do
 begin
    s:=0;
   {==нахождение суммы делителей==}
```

```
   for i:=1 to (n div 2) do
        if n mod i=0 then s:=s+i;
    if s=n then writeln (n);
 end;
end.
```

Тест:

Дано:	0,100
Результат:	6 28

Задача: Проверить, являются ли два числа, введенных с клавиатуры дружественными.

Программа на Бейсике:

```
input "введите два числа"; n, m
rem=========нахождение суммы  делителей n==================
for i=1 to n/2
   if n mod i=0 then s1=s1+i
next
rem=========нахождение суммы  делителей m==================
for i=1 to m/2
   if m mod i=0 then s2=s2+i
next
if (n=s2) and (m=s1) then print "дружественные" else print "не дружественные"
```

Программа на Паскале:

```
var n,m,i,s1,s2: integer;
begin
 writeln ('введите два числа');
 readln (n,m);
 s1:=0;
 s2:=0;
 {=нахождение суммы  делителей n=}
 for i:=1 to (n div 2) do
    if n mod i=0 then s1:=s1+i;
 {=нахождение суммы  делителей m=}
 for i:=1 to (m div 2) do
    if m mod i=0 then s2:=s2+i;
 if (s2=n) and (s1=m) then writeln ('числа дружественные')
 else writeln ('числа не дружественные');
end.
```

Тест:

Дано:	220, 284
Результат:	Числа дружественные

Задача: Ввести строку. Найти ее период.

Идея решения: Необходимо найти К (количество символов в строке), определить делители К и сравнить вырезки из строки по R символов (R-делитель К)

Программа на Бейсике:

```
input "введите строку"; a$
n=len (a$)
for i=n to 1 step -1
 k=1
 if n mod i=0 then
 rem=вырезка по i====================
 for j=1 to n step i
     if mid$ (a$, j, i)=mid$(a$,j+i,i) then k=k+1
 next
 if k=n/i then kol=i
 end if
next
print mid$ (a$,1,kol)
```

Программа на Паскале:

```
var n,i,j,k,x,h,period: integer;
 a:string;
begin
 writeln ('введите строку');
 readln (a);
 n:=length(a);
 for i:=1 to n do
 begin
     k:=0;
    if n mod i=0 then
        begin
        h:=n div i;
        x:=1;
        for j:=1 to i-1 do
            begin
            if copy (a,1,h)=copy(a,x+h,h) then k:=k+1;
            x:=x+h;
            end;
        if k=i-1 then period:=h;
       end;
  end;
  writeln (copy (a, 1, period));
end.
```

Тест:

Дано:	информатикаинформатикаинформатика
Результат:	информатика

Краткие итоги лекции сформулированы в таблице 1.2:

Таблица 1.2.

	Бейсик	**Паскаль**

Зависимость уменьшающейся переменной X в теле цикла от увеличивающегося значения счетчика цикла i	`for i=1 to n`
	`x=n-i+1`
	`next`
		`for i:=1 to n do`
		`x:=n-i+1;`

Признак кратности	`1. x \ y = x / y` `2. x mod y =0`	`x mod y = 0`

Длина отрезка	`sqr ((x1-x2)^2+(y1-y2)^2)`	`sqrt (sqr(x1-x2)+sqr(y1-y2))`

Ключевые термины

- Счетчик цикла - переменная, в которой хранится номер шага выполнения тела цикла.
- Палиндром - запись, читаемая одинаково слева направо и справа налево.
- Кратность - делимость одного числа на другое без остатка. Арифметическая операция mod дает в результате остаток от деления двух чисел друг на друга.
- Гипотенуза прямоугольного треугольника - находится по теореме Пифагора как квадратный корень суммы квадратов катетов.

Набор для практики

Вопросы.

1. Какова зависимость уменьшающейся переменной X в теле цикла от увеличивающегося значения счетчика цикла i, если счетчик работает от 0 до 10?
2. Сформулируйте признак кратности числа X числу У, признак четности числа. Как определить "круглое" ли число?
3. Запишите арифметическое выражение, в котором вычисляется длина отрезка, если известны координаты его концов.

Упражнения из раздела "Графика на Паскале".

Задача: Написать программу для движения точки (или любого другого графического объекта) по четырем направлениям, показанным на рис. 1.3:

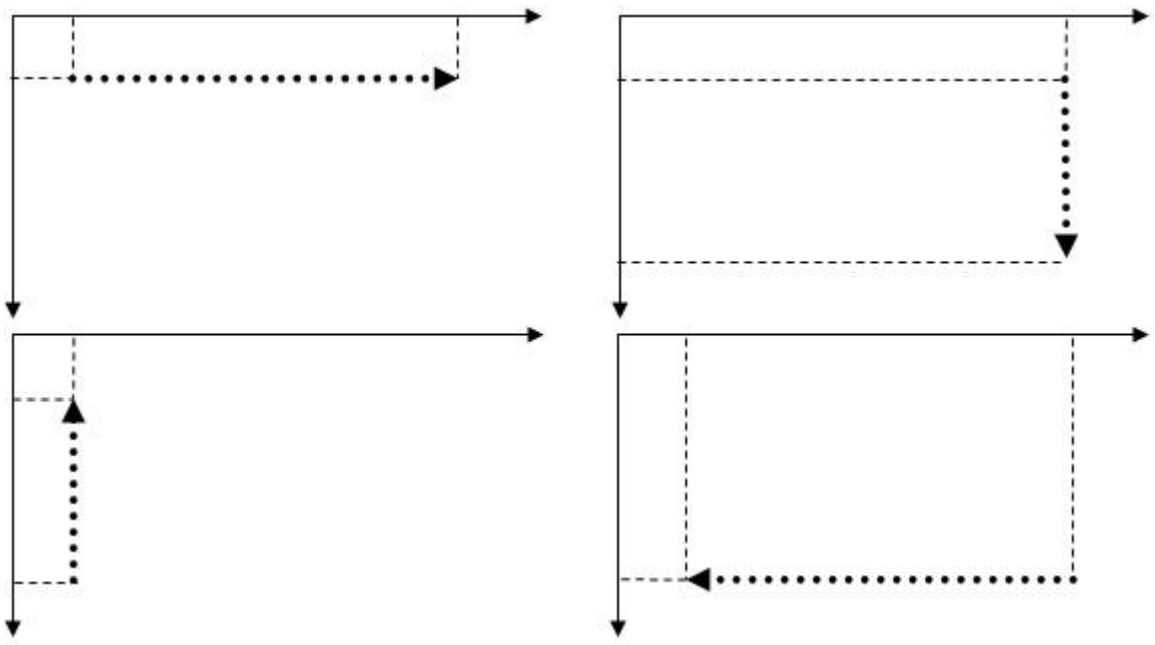

Рис. 1.3. Траектория движения точки

Задача: Написать программу для движения отрезка по экрану монитора. Отрезок "скользит" по горизонтальным краям окна в направлениях, показанных на рис. 1.4:

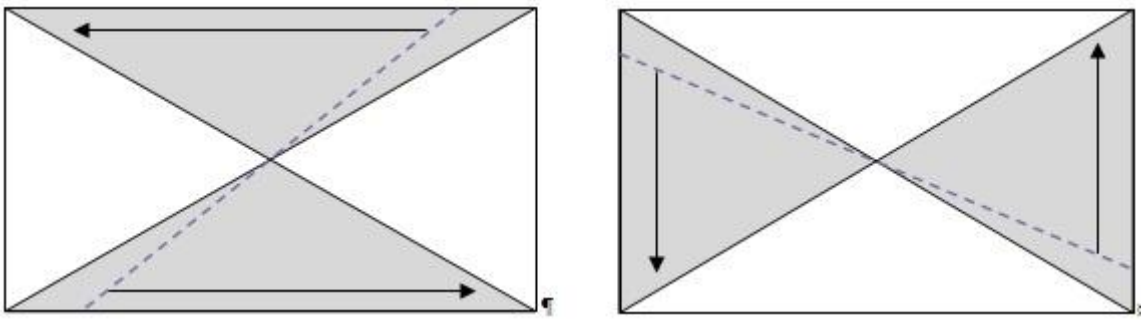

Рис. 1.4. Траектория движения отрезка

Используя типовые алгоритмы можно решить любую задачу. В лекции очерчен круг НЕОБХОДИМЫХ ТИПОВЫХ АЛГОРИТМОВ (для обработки одномерных массивов и обработки строк), рассмотрены некоторые олимпиадные задачи, которые решаются с использованием этих алгоритмов. **Цель лекции:** научиться применять изученные типовые алгоритмы при решении классических задач.

Содержание

Типовые алгоритмы обработки одномерных массивов

В данной теме будут рассматриваться такие типовые алгоритмы обработки одномерных массивов:

- Заполнение, вывод элементов массива
- Сумма, произведение элементов
- Выбор по условию
- Максимальный (минимальный) элемент
- Вставка, удаление элементов
- Инвертирование (изменения порядка следования элементов заданного массива на обратный)

Программные реализации типовых алгоритмов обработки одномерных массивов приведены в таблице 2.1:

Таблица 2.1.

Типовой алгоритм	Программная реализация (Бейсик)	Программная реализация (Паскаль)

Заполнение массива	`input n`
	`dim a(n)`
	`for i=1 to n`
	`input a(i)`
	`next i`
		`const n=10;`
		`Var a: array [1..n] of integer;`
		`begin`
		`for i:=1 to n do readln (a[i]);`
		`…`

Вывод в строку	`…for i=1 to n` `print a(i) ; "  " ;`	`…` `for i:=1 to n do write`

	next i	(a[i]); …

Сумма, произведение элементов	…
	p=1
	for i=1 to n
	s=s + a(i)
	p=p * a(i)
	next i
		…
		s:=0; p:=1;
		for i:=1 to n do
		begin
		s:=s+a[i]);
		p:=p*a[i]);
		end;
		…

Выбор по условию	…
	p = 1
	for i = 1 to n
	if {условие} then
	k=k+1:s=s+a(i):p=p*a(i)
	next i
		…
		k:=0; s:=0; p:=1;
		for i:=1 to n do
		if {условие} then
		begin
		k:=k+1; s:=s+a[i];
		p:=p*a[i];
		end;
		…

Максимальный (минимальный) элемент	…
	max = a(1): min = a(1)
	for i = 2 to n
	if a(i) > max then max = a(i)
	if a(i) < min then min = a(i)
	next i
		…
		max:=a[1]; min:=a[1];
		for i:=1 to n do
		begin
		if a[i] > max then
		max:=a[i];
		if a[i] < min then
		min:=a[i];
		end;

Вставка x на k-ое место	dim a(n + 1)
	…
	for i = n to k step -1
	a (i + 1) = a(i)
	next
	a(k) = x
		Var a: array [1..n+1] of…
		…
		for i:=n downto k do
		a[i+1]:=a[i];
		a[k]:=x;
		…

Удаление k-ого элемента	. . .
	for i = k to n - 1
	a(i) = a(i + 1)
	next
		…
		for i:=k to (n-1) do
		a[i]:= a[i+1];
		…

Инвертирование элементов	. . .
	for i = 1 to n/2
	swap a(i), a(n-i+1)
	next
		…
		for i:=1 to (n div 2) do
		begin
		x:=a[i];
		a[i]:= a[n-i+1];
		a[n-i+1] :=x;
		end
		…

Ключевые моменты в типовых алгоритмах:

- Выбор по условию. В качестве условия может проверяться значение элемента массива на четность, кратность элемента какому-либо числу, положительность, отрицательность, равенство нулю. Может проверяться также и значение индекса элемента массива (например, элементы, стоящие на четных местах и др.).
- Максимальный (минимальный) элемент. Кроме максимального элемента часто требуется найти и индекс максимального элемента:
- `if a[i]>max then begin`
- `max:=a[i]; imax:=i;`

```
end;
```

- Вставка x на k-ое место. Перестановка элементов (для освобождения "места" для вставляемого элемента) происходит с конца массива - последний элемент передвигается на "пустое место", на его место передвигается предпоследний элемент и т.д.
- Инвертирование элементов. Цикл работает n/2 раз, так как за один проход мы меняем сразу два элемента местами.

Задачи использованием типовых алгоритмов обработки одномерных массивов

Задача: На плоскости изображено N прямоугольников (рис. 2.1). Каждый прямоугольник задан координатами левой нижней и правой верхней вершин. Определить, имеют ли прямоугольники общую площадь .

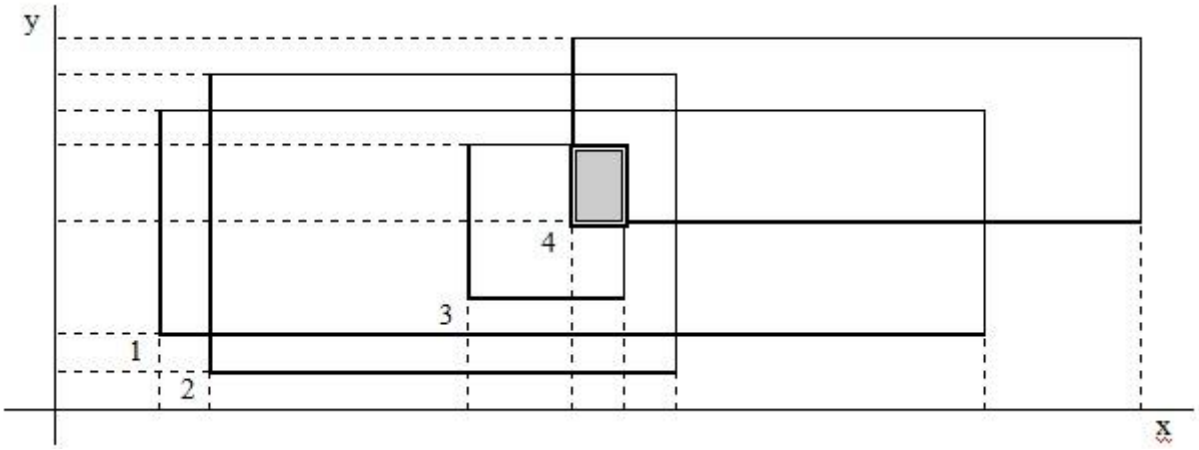

Рис. 2.1.

Идея решения:

Если:

- максимальная координата по оси X левых нижних вершин прямоугольников будет меньше минимальной координаты правых верхних вершин и …
- …максимальная координата по оси У левых нижних вершин прямоугольников будет меньше минимальной координаты правых верхних вершин, то …
- …общая площадь есть.

В задаче необходимо использовать типовой алгоритм нахождения МАКСИМАЛЬНОГО (МИНИМАЛЬНОГО) ЭЛЕМЕНТА МАССИВА.

Для вычисления общей площади необходимо найти произведение разности:

- максимальной координаты по оси X левых нижних вершин прямоугольников и минимальной координаты правых верхних вершин и …
- …максимальной координаты по оси У левых нижних вершин прямоугольников и минимальной координаты правых верхних вершин.

Программа на Бейсике:

```
input "введите количество прямоугольников"; n
dim x1(n), x2(n),y1(n), y2(n)
for i=1 to n
    input x1(i), x2(i), y1(i), y2(i)
next
xmax=x1(1)
xmin=x2(1)
ymax=y1(1)
ymin=y2(2)
for i=1 to n
    if x1(i) > xmax then xmax=x1(i)
    if x2(i) < xmin then xmin=x2(i)
    if y1(i) > ymax then ymax=y1(i)
    if y2(i) < ymin then ymin=y2(i)
next
if xmax<xmin and ymax<ymin then print "общ. площадь есть" else print "общ.
площади нет"
```

Программа на Паскале:

```
var     x1, x2, y1, y2: array [1..10] of integer;
    n, i, xmax, xmin, ymax, ymin: integer;
begin
    writeln ('введите количество прямоугольников');
    readln (n);
    for i:=1 to n do
        readln (x1[i], y1[i], x2[i], y2[i]);
    xmax:=x1[1];
    xmin:=x2[1];
    ymax:=y1[1];
    ymin:=y2[2];
    for i:=1 to n do
        begin
        if x1[i] > xmax then xmax:=x1[i];
        if x2[i] < xmin then xmin:=x2[i];
        if y1[i] > ymax then ymax:=y1[i];
        if y2[i] < ymin then ymin:=y2[i];
        end;
    if (xmax<xmin) and (ymax<ymin) then writeln ('общая площадь есть')
    else writeln ('общей площади нет');
end.
```

Тест:

Дано:	3 1,1,9,5 2,3,5,6 4,2,7,4	4 2,2,5,4 4,3,7,6 6,1,11,2 6,4,12,8
Результат:	Общая площадь есть	Общей площади нет

Задача: Латинским квадратом называется массив, в строках и столбцах которого нет одинаковых элементов. Вывести на экран латинский квадрат размером NxN.

Идея решения: Заполнить 1 строку квадратного массива (NxN) числами от 1 до N. Вторая строка массива получается путем циклического сдвига элементов первой строки и т. д. (табл. 2.2). Циклический сдвиг можно реализовать, используя типовой алгоритм ВСТАВКИ-УДАЛЕНИЯ (в зависимости от направления циклического сдвига).

Таблица 2.2. Заполнение латинского квадрата путем циклического сдвига элементов

1	2	3	4	5
5	1	2	3	4
4	5	1	2	3
3	4	5	1	2
2	3	4	5	1

Решение на Бейсике:

```
input "размерность="; n
dim a(n,n)
for j=1 to n
    a(1,j)=j
next
rem=====сдвиг=========
for i=2 to n
    for j=1 to n
        a(i,j)=a(i-1,j)
    next
    x=a(i,n)
    for j=n to 2 step -1
        a(i,j)=a(i,j-1)
    next
    a(i,1)=x
next
rem====вывод==========
for i=1 to n
    for j=1 to n
       print  a(i,j);
    next
    print
next
```

Решение на Паскале:

```
var a: array [1..10,1..10] of integer;
    n,i,j,x: integer;
begin
    writeln ('размерность=');
    readln (n);
    for j:=1 to n do a[1,j]:=j;
    {======сдвиг======}
    for i:=2 to n do
        begin
        for j:=1 to n do a[i,j]:=a[i-1,j];
        x:=a[i,n];
        for j:=n downto 2 do
            a[i,j]:=a[i,j-1];
        a[i,1]:=x;
        end;
    {======вывод======}
    for i:=1 to n do
        begin
```

```
for j:=1 to n do write(a[i,j]);
writeln;
end;
```

Типовые алгоритмы обработки строковых переменных

В теме рассматриваются такие типовые алгоритмы обработки строк, которые помогут:

- "Разобрать" число на цифры, поместив каждую цифру в ячейку массива.
- "Разобрать" строку, поместив каждый символ в ячейку массива.
- "Разобрать" предложение, поместив каждое слово в ячейку массива.

Программные реализации данных алгоритмов приведены в таблице 2.3:

Таблица 2.3.

Типовой алгоритм	Программная реализация

"Разобрать" строки на буквы	Бейсик:
	
	`input a$`
	`n=len(a$)`
	`dim a$(n)`
	`for i=1 to n`
	`a$(i)= mid$(a$, i, 1)`
	`next`
	
	Паскаль:
	
	`var a: array [1..10] of char;`
	`stroka: string;`
	`i, n:integer;`
	`begin`
	`readln (stroka);`
	`n:=length(stroka);`
	`for i:=1 to n do`
	`a[i]:=copy(stroka,i,1);`
	`…`
	`end.`

"Разобрать" число на цифры	Бейсик:
	
	`input a$`
	`n=len(a$)`
	`dim a(n)`
	`for i=1 to n`
	`a(i)=val (mid$(a$,i,1))`
	`next`
	
	Паскаль:
	
	`Var a: array [1..10] of byte;`
	`stroka: string;`
	`I, n, k:integer;`
	`begin`
	`readln (stroka);`
	`n:=length(stroka);`
	`for i:=1 to n do`
	`val(copy(stroka,i,1),a[i], к);`
	`…`

	end.

"Разобрать" предложение на слова	Бейсик:
	
	input a$
	n=len(a$)
	for i=1 to n
	if mid$(a$,i,1)=" " then k=k+1
	next
	k=k+1 {слов}
	dim a$(k)
	j=1
	for i=1 to n
	if mid$(a$,i,1)=" " then j=j+1 else
	a$(j)=a$(j)+mid$(a$,i,1)
	next
	
	Паскаль:
	
	var a: array [1..20] of string;
	stroka: string;
	i, n, j,k:integer;
	begin
	readln (stroka);
	n:=length(stroka);
	j:=1;
	for i:=1 to n do
	if copy(stroka,i,1)=' ' then j:=j+1
	else a[j]:=a[j]+copy(stroka,i,1);
	…
	end.

Решение задач с использованием типовых алгоритмов обработки строковых переменных

Задачи "Разбор числа на цифры"

Идея решения: Во всех задачах данной группы число необходимо ввести в строковую ячейку. Затем, применив типовой алгоритм "РАЗБОРА" ЧИСЛА НА ЦИФРЫ, произвести необходимые действия. Ячейка, в которой накапливается произведение в цикле, должна быть предварительно заполнена единицей.

Ввести число. Найти сумму, произведение цифр числа.

Программа на Бейсике:

```
input "введите число "; a$
n=len(a$)
for i=1 to n
    s=s+val (mid$(a$,i,1))
    p=p*val (mid$(a$,i,1))
next
print s
```

Программа на Паскале:

```
var a: string;
    n,s,i,x,k:integer;
begin
```

```
    writeln ('введите число');
    readln (a);
    n:=length(a);
    s:=0; p:=1;
    for i:=1 to n do
        begin
        val (copy(a,i,1),x,k);
        s:=s+x; p:=p*x;
        end;
    writeln (s, p);
end.
```

Тест:

Дано:	1234
Результат:	10 24

1. Ввести число. Найти максимальную цифру числа

   Программа на Бейсике:

   ```
   input "введите число"; a$
   n=len(a$)
   max=val (mid$(a$,1,1))
   for i=1 to n
       if val (mid$(a$,i,1))> max then max=val (mid$(a$,i,1))
   next
   print max, min
   ```

   Программа на Паскале:

   ```
   var a: string;
       n,i,x,k,max: integer;
   begin
       writeln ('введите число'); readln (a);
       n:=length(a);
       val (copy(a,1,1),x,k);
       max:=x;
       for i:=1 to n do
           begin
           val (copy(a,i,1),x,k);
           if x>max then max:=x;
           end;
       writeln ('max=',max);
   end.
   ```

   Тест:

Дано:	183058436
Результат:	8

2. Ввести число. Найти количество нулевых цифр в числе

   Программа на Бейсике:

   ```
   input "введите число"; a$
   n=len (a$)
   ```

```
for i=1 to n
    if val (mid$(a$,i,1))=0 then k=k+1
next
print k
```

Программа на Паскале:

```
var a: string;
    n,i,x,k,kk: integer;
begin
    writeln ('введите число');
    readln (a);
    n:=length (a);
    k:=0;
    for i:=1 to n do
        begin
        val (copy(a,i,1),x,kk);
        if x=0 then k:=k+1;
        end;
    writeln ('k=', k);
end.
```

Тест:

Дано:	10500304500
Результат:	K=6

3. Определить, является ли число, введенное с клавиатуры числом Армстронга. Число Армстронга: сумма цифр этого числа, возведенных в n-ую степень (n - количество цифр числа) равна самому числу.

   Идея решения: Число необходимо ввести в строковую ячейку. Аналогично предыдущей задаче необходимо применить типовой алгоритм РАЗБОРА ЧИСЛА НА ЦИФРЫ. В программе на Паскале возведение числа в n-ую степень реализовано в цикле.

   Бейсик:

```
input "введите число"; a$
n = len (a$)
for  i=1 to  n
    s=s+val(mid$ (a$,i,1))^n
next
if s=val(a$) then print "это число Армстронга" else print "это не число
Армстронга"
```

   Паскаль:

```
    var a: string;
    b, i, j, st, n, s, x, k: integer;
begin
    writeln ('введите число');
    readln (a);
    n:=length(a);
    s:=0;
    for i:=1 to n do
        begin
        val (copy(a,n-i+1,1),x,k);
```

```
        {=возведение числа в степень=}
        st:=1;
        for j:=1 to n do
            st:=st*x;
        {====================}
        s:=s+st;
        end;
    val (a, x, k);
    if s=x then writeln ('это число Армстронга')
    else writeln ('это не число Армстронга');
end.
```

Тест:

Дано:	153	1634
Результат:	Это число Армстронга	Это число Армстронга

4. Найти цифровой корень числа. **Цифровой корень** числа получается при сложении цифр числа, затем при сложении цифр вновь полученного числа и так до тех пор, пока в сумме не будет получена одна цифра.

   Идея решения: Решение задачи также базируется на типовом алгоритме "РАЗБОРА" ЧИСЛА НА ЦИФРЫ. В решении задачи на Бейсике логическое выражение в цикле While выглядит как Len(A$)>2, а не >1, т.к. функция STR$ записывает один лишний пробел перед словом.

   Программа на Бейсике:

```
input "введите число"; a$
while len (a$) > 2
    for i=1 to len (a$)
        s=s+val (mid$ (a$, i, 1))
    next
    a$=str$ (s)
    s=0
wend
print"цифровой корень числа="; a$
```

   Программа на Паскале:

```
var a: string;
    n,s,i,x,k:integer;
begin
    writeln ('введите число');
    readln (a);
    n:=length(a);
    while n>1 do
        begin
        s:=0;
        for i:=1 to n do
            begin
            val (copy(a,i,1),x,k);
            s:=s+x;
            end;
        str(s,a);
        n:=length(a);
        end;
        writeln ('цифровой корень числа=', s);
end.
```

Тест:

Дано:	123456
Результат:	3

5. Перевести число из одной системы счисления в другую.

   Идея решения: перевод числа в десятичную систему счисления базируется на типовом алгоритме "РАЗБОРА" ЧИСЛА НА ЦИФРЫ. Перевод из десятичной системы счисления не опирается на рассмотренные выше типовые алгоритмы, но мы приведем и его.

Перевод числа из B-ричной (B<10) в десятичную систему счисления:	Бейсик:
	
	`input "введите число и основание c.c."; a$, b`
	`n = len (a$)`
	`for  i = 1 to  n`
	`s=s+val(mid$ (a$,i,1)) *b^(n - i)`
	`next`
	`print s`
	
	Паскаль:
	
	`var a: string;`
	`b, i, st, n, s, x, k:integer;`
	`begin`
	`readln (a); readln(b);`
	`n:=length(a);`
	`s:=0;`
	`st:=1;`
	`for i:=1 to n do`
	`begin`
	`val (copy(a,n-i+1,1),x,k);`
	`s:=s+x*st;`
	`st:=st*b;`
	`end;`
	`writeln (s);`
	`end.`
	
	Тест:
	
	Дано: 1101, 2
	Результат: 13

Перевод числа из десятичной системы счисления в B-ричную	Бейсик:
	
	`input "введите число и основание системы"; a, b`
	`while  a<>0`
	`x = a mod b`
	`s = s * 10 + x`
	`a = a \ b`
	`wend`
	`print s`

Паскаль:

```
var a, b, s, x: integer;
begin
readln (a); readln (b);
s:=0;
while a<>0 do
begin
x:=a mod b;
s:=s*10 +x;
a:=a div b;
end;
writeln (s);
end.
```

Тест:

Дано:	13, 2
Результат:	1101

Задачи "Разбор предложения на слова"

1. Заменить слово Slovo1 на слово Slovo2 в предложении. Вывести новое предложение на экран.

   Решение задачи на Бейсике:

   ```
   input "введите предложение"; a$
   input "искомое слово"; slovo1$
   input "слово-замена"; slovo2$
   n=len(a$)
   for i=1 to n
       if mid$(a$,i,1)=" " then k=k+1
   next
   k=k+1   dim a$(k)
   j=1
   for i=1 to n
       if mid$(a$,i,1)=" " then j=j+1 else a$(j)=a$(j)+mid$(a$,i,1)
   next
   for i=1 to k
       if a$(i)=slovo1$ then a$(i)=slovo2$
   next
   for i=1 to k
       print a$(i); " ";
   next
   ```

   Решение задачи на Паскале:

   ```
   var b: array [1..10] of string;
       a, a1, a2: string;
       n, i, j, k: integer;
   begin
       writeln ('предложение'); readln (a);
       writeln ('искомое слово'); readln (a1);
       writeln ('слово-замена'); readln(a2);
       n:=length (a);
       k:=0;
       for i:=1 to n do
   ```

```
        if copy(a,i,1)=' ' then k:=k+1;
    k:=k+1;
    j:=1;
    for i:=1 to n do
        if copy(a,i,1)=' ' then j:=j+1
        else b[j]:=b[j]+copy(a,i,1);
    for i:=1 to k do
        if b[i]=a1 then b[i]:=a2;
    for i:=1 to k  do
        writeln (b[i]);
end.
```

Тест:

Дано:	Мама мыла раму мыла красила
Результат:	Мама красила раму

2. Вывести на экран самое длинное и самое короткое слово в предложении.

   Идея решения: Введенное с клавиатуры предложение, воспользовавшись типовым алгоритмом "РАЗБОРА" ПРЕДЛОЖЕНИЯ НА СЛОВА помещаем в массив. Затем, применив типовой алгоритм ПОИСКА МАКСИМАЛЬНОГО ЭЛЕМЕНТА найдем самое длинное и самое короткое слово в предложении.

   Решение задачи на Бейсике:

```
input "введите предложение"; a$
n=len(a$)
for i=1 to n
    if mid$(a$,i,1)=" " then k=k+1
next
k=k+1
dim a$(k)
j=1
for i=1 to n
    if mid$(a$,i,1)=" " then j=j+1 else a$(j)=a$(j)+mid$(a$,i,1)
next
min=len(a$(1))
max=len(a$(1))
for i=1 to k
    if len (a$(i))>max then max=len (a$(i)): nmax=i
    if len (a$(i))<min then min=len (a$(i)): nmin=i
next
print "самое короткое слово-"; a$(nmin)
print "самое длинное слово-"; a$(nmax)
```

   Решение задачи на Паскале:

```
var a: array [1..100] of string;
    b: string;
    nmax, nmin, max, min, n, s, i, j, x, k: integer;
begin
    readln (b);
    n:=length(b);
```

```
    k:=0;
    for i:=1 to n do
        if copy(b,i,1)=' ' then k:=k+1;
    k:=k+1;
    j:=1;
    for i:=1 to n do
        if copy(b,i,1)=' ' then j:=j+1
        else a[j]:=a[j]+copy(b,i,1);
    min:=length(a[1]);
    max:=length(a[1]);
    for i:=1 to k do
        begin
        if length(a[i])>max then
            begin
            max:=length (a[i]);
            nmax:=i;
            end;
        if length(a[i])<min then
            begin
            min:=length (a[i]);
            nmin:=i;
            end;
            end;
    writeln ('самое короткое слово-', a[nmin]);
    writeln ('самое длинное слово-', a[nmax]);
end.
```

Тест:

Дано:	Aaaa ss dddd fffff g hh
Результат:	самое короткое слово-g
	самое длинное слово-fffff

3. Ввести предложение. Выдать его на экран, поменяв порядок следования слов в предложении.

   Идея решения: Введенное с клавиатуры предложение, воспользовавшись типовым алгоритмом "РАЗБОРА" ПРЕДЛОЖЕНИЯ НА СЛОВА помещаем в массив. Выводим массив с конца. Для организации хранения в массиве слов в обратном порядке необходимо воспользоваться зависимостью УМЕНЬШАЮЩЕЙСЯ ПЕРЕМЕННОЙ ОТ УВЕЛИЧИВАЮЩЕГОСЯ СЧЕТЧИКА ЦИКЛА и поменять порядок следования слов в массиве (проделайте это самостоятельно).

   Решение задачи на Бейсике:

```
input "введите предложение"; a$
n=len(a$)
for i=1 to n
    if mid$(a$,i,1)=" " then k=k+1
next
k=k+1
dim a$(k)
j=1
for i=1 to n
    if mid$(a$,i,1)=" " then j=j+1 else a$(j)=a$(j)+mid$(a$,i,1)
next
for i= k to 1 step -1
    print a$(i); " ";
```

```
next
```

Решение задачи на Паскале:

```
var b: array[1..10] of string;
    a: string;
    n, i, j, k: integer;
begin
    writeln ('предложение');
    readln (a);
    n:=length (a);
    k:=0;
    for i:=1 to n do
        if copy(a,i,1)=' ' then k:=k+1;
    k:=k+1;
    j:=1;
    for i:=1 to n do
        if copy(a,i,1)=' ' then j:=j+1
    else b[j]:=b[j]+copy(a,i,1);
    for i:=k downto 1 do
        writeln (b[i]);
end.
```

Тест:

Дано:	Мама мыла раму
Результат:	раму мыла Мама

4. Ввести предложение. Выдать его на экран, изменив порядок следования букв в каждом слове, оставив порядок следования слов в предложении прежним.

   Идея решения: Введенное с клавиатуры предложение, воспользовавшись типовым алгоритмом "РАЗБОРА" ПРЕДЛОЖЕНИЯ НА СЛОВА помещаем в массив. Инвертируем содержимое каждого элемента массива. В приведенной ниже программе на Паскале в массиве хранится предложение с обратным порядком слов. Изменить порядок слов можно, воспользовавшись зависимостью УМЕНЬШАЮЩЕЙСЯ ПЕРЕМЕННОЙ ОТ УВЕЛИЧИВАЮЩЕГОСЯ СЧЕТЧИКА ЦИКЛА.

   Решение задачи на Бейсике:

```
input "введите предложение"; a$
n=len(a$)
for i=1 to n
    if mid$(a$,i,1)=" " then k=k+1
next
k=k+1
dim a$(k)
j=1
for i=1 to n
    if mid$(a$,i,1)=" " then j=j+1 else a$(j)=a$(j)+mid$(a$,i,1)
next
rem=инвертируем слова=========================
for i= 1 to k
    x$=" "
    for j=len (a$(i)) to 1 step -1
        x$=x$+mid$ (a$(i),j,1)
    next j
```

```
    a$(i)=x$
next i
for i= 1 to k
    print a$(i); " ";
next
```

Решение задачи на Паскале:

```
var b: array[1..10] of string;
    a: string;
    n, i, j, k: integer;
begin
    writeln ('предложение'); readln (a);
    n:=length (a);
    k:=0;
    for i:=1 to n do
        if copy(a,i,1)=' ' then k:=k+1;
    k:=k+1;
    j:=1;
    for i:=n downto 1 do
        if copy(a,i,1)=' ' then j:=j+1
        else b[j]:=b[j]+copy(a,i,1);
    for i:=k downto 1 do
        writeln (b[i]);
end.
```

Тест:

Дано:	Мама мыла раму
Результат:	амаМ алым умар

Ключевые термины

- Одномерный массив - именованный набор однотипных переменных, расположенных в памяти непосредственно друг за другом, доступ к которым осуществляется по индексу.
- Латинский квадрат - двумерный массив, в каждой строке и столбце которого нет повторяющихся элементов.
- Циклический сдвиг элементов массива - сдвиг всех элементов массива вправо (влево) с помещением последнего элемента на первое место (первого элемента на последнее место).
- Число Армстронга - сумма цифр этого числа, возведенных в n-ую степень (n - количество цифр числа) равна самому числу.
- Цифровой корень числа - получается при сложении цифр числа, затем при сложении цифр вновь полученного числа и так до тех пор, пока в сумме не будет получена одна цифра.
- Система счисления - символический метод записи чисел, представление чисел с помощью письменных знаков.

Краткие итоги

Необходимые для решения задач типовые алгоритмы обработки одномерных массивов:

- Заполнение, вывод элементов массива
- Сумма, произведение элементов
- Выбор по условию

- Максимальный (минимальный) элемент
- Вставка, удаление элементов
- Инвертирование элементов

Необходимые типовые алгоритмы обработки строк:

- "Разбор" числа на цифры, помещение каждой цифры в ячейку массива.
- "Разбор" строки, помещение каждого символа в ячейку массива.
- "Разбор" предложения, помещение каждого слова в ячейку массива.

Набор для практики

Вопросы.

- Какова зависимость индексов элементов, которые мы меняем местами при выполнении инвертирования элементов массива от счетчика цикла?
- Что произойдет, если в типовом алгоритме инвертирования элементов массива счетчик цикла отработает до n (n - количество элементов в массиве)?
- Что произойдет, если в типовом алгоритме вставки элементов перемещение элементов с i-ной в (i+1)-ую позицию производить не с конца массива, а начиная с номера вставляемого элемента?
- Каково назначение функций length (a) и copy(a,k,n)?
- Сформулируйте алгоритм перевода числа из n-ричной системы счисления в 10-ую.
- Сформулируйте алгоритм перевода числа из 10-тичной системы счисления в n-ричую.

Упражнения

- Найти произведение четных, сумму отрицательных, количество нулевых элементов одномерного массива размерностью 10, заполненного с клавиатуры.
- Удалить максимальный элемент из одномерного массива размерностью 10, заполненного с клавиатуры. Вставить после минимального элемента ноль.
- Ввести число. Найти произведение четных цифр в нем.
- Ввести число в десятичной системе счисления. Определить, чего больше - нулей или единиц в его двоичном представлении?
- Ввести предложение. Найти, сколько слов нем начинается и заканчивается одной и той же буквой.

Часто на олимпиадах предлагаются задачи, в которых необходимо вычислить результат арифметических операций над сверхбольшими числами. При решении этих задач мы будем опираться на рассмотренные в предыдущей лекции типовые алгоритмы обработки одномерных массивови строк. **Цель лекции:** научиться производить операции со "сверхбольшими" числами при решении классических задач.

Содержание

Архитектура 32-х разрядных систем позволяет обрабатывать числа в максимальном диапазоне 0..4294967295. Но это слишком узкий диапазон натуральных чисел для решения многих прикладных задач (см. табл. 3.1).

Справочные сведения:

Таблица 3.1. Форматы целых чисел

Описатель типа	Длина(байт)	Минимальное число	Максимальное число
Integer	2 (со знаком)	-32768	+32767
Shortint	1 (со знаком)	-128	+127
Longint	4 (со знаком)	-2147483648	+2147483647
Byte	1 (без знака)	0	255
Word	2 (без знака)	0	65535

Учиться производить операции со "сверхбольшими" числами будем на практике.

Задача: Найти произведение сверхбольшого числа на цифру.

Идею решения рассмотрим на примере, в котором нужно найти произведение "сверхбольшого" числа 4510905723598 на цифру 3:

- Вводим число в СТРОКОВУЮ переменную.
- "РАЗБИРАЕМ" ЧИСЛО НА ЦИФРЫ, помещая каждую цифру в элемент массива:

  4 5 1 0 9 5 7 2 3 5 9 8

- Умножаем каждый элемент на "3":

  12 15 3 0 27 0 15 21 6 9 15 27 24

- Организуем перенос: в каждой ячейке оставляем младшую цифру хранящегося там числа, а старшую цифру суммируем с числом, находящимся в левой ячейке:

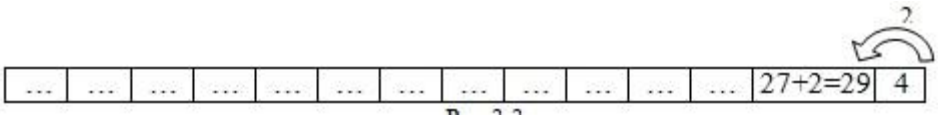

Рис. 3.1.

Программа на Бейсике:

```
input  "введите длинное число"; a$
n = len(a$)
dim a(n), rez(n)
input  "второй сомножитель"; x
for i=1 to n
  a(i)=val(mid$(a$, i, 1))
next
rem=========================
for i=1 to n
  rez(i)=a(i) * x
next
for i=n to 2 step -1
  rem==перенос в старший разряд==
  rez(i - 1) = rez(i - 1) + (rez(i) \ 10)
  rem==остаток в младшем разряде=
  rez(i) = rez(i) mod 10
next
rem====вывод результата==========
for i = 1 to n
  print rez(i);
next
```

Программа на Паскале:

```
const   m=100;
var a, rez: array [1..m] of byte;
  i, n, x, k: integer;
  stroka: string;
begin
  writeln  ('введите длинное число'); readln (stroka);
  n:= length (stroka);
  writeln ('второй сомножитель'); readln (x);
  for i:=1 to n do
    val (copy(stroka, i, 1), a[i], k);
  {=========================}
  for i:=1 to n do
    rez[i]:= a[i] * x;
  for i:=n downto 2 do
    begin
    rez[i - 1]:= rez[i - 1] + rez[i] div 10;
        rez[i]:= rez[i] mod 10;
        end
  {====вывод результата==========}
  for i:=1 to n do
        write (rez[i]);
end.
```

Тест:

Дано:	32859305092145 5
Результат:	164296525460725

Задача: Найти сумму двух "сверхбольших" чисел.

Идею решения рассмотрим на примере вычисления суммы двух "сверхбольших" чисел 4510905723569 и 361295487.

Вводим числа в строковые переменные. "РАЗБИРАЕМ" каждое ЧИСЛО НА ЦИФРЫ, помещая цифры в элементы массивов. Массив, в котором будет храниться меньшее число (в примере это массив b) будем заполнять с 5-ой ячейки.

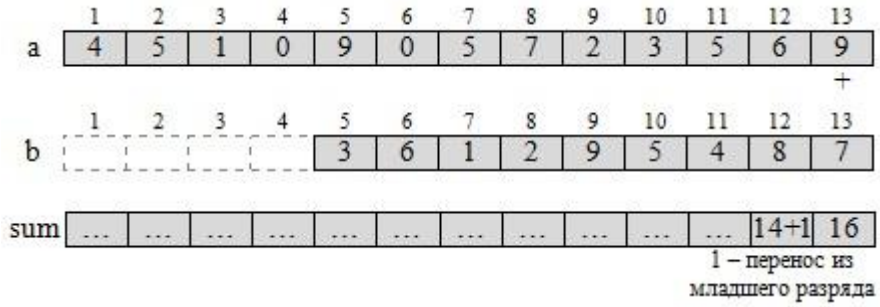

Рис. 3.2.

Массив Sum заполняется суммой соответствующих ячеек массивов A и B. Затем организуем перенос: в каждой ячейке оставляем младшую цифру хранящегося там числа, а старшую цифру суммируем с числом, находящимся в левой ячейке.

Программа на Бейсике:

```
input "первый сомножитель"; a$
input "второй сомножитель"; b$
n = len(a$)
m = len(b$)
if n > m then max = n else max = m
dim a(max), b(max), sum(max)
rem=========================
for i = 1 to n
  a(i + (max - n)) = val(mid$(a$, i, 1))
next
for i = 1 to m
  b(i + (max - m)) = val(mid$(b$, i, 1))
next
rem=========================
for i = 1 to max
  sum(i) = a(i) + b(i)
next
for i = max to 2 step -1
  sum(i - 1) = sum(i - 1) + sum(i) \ 10
  sum(i) = sum(i) mod 10
next
rem=========================
for i = 1 to max
  print (sum(i));
```

Программа на Паскале:

```
const   mm=100;
var a,b, sum: array [1..mm] of byte;
  i, n, m,max, x, k: integer;
  strokaA, strokaB: string;
begin
  writeln ('первый сомножитель'); readln  (strokaA);
  writeln ('второй сомножитель'); readln (strokaB);
  n:= length (strokaA);
  m:= length (strokaB);
  if n>m then max:=n
    else max:=m;
  for i:=1 to n do
    begin
    val(copy(strokaA, i, 1),x,k);
        a[i+(max-n)]:=x;
        end;
  for i:=1 to m do
    begin
        val(copy(strokaB, i, 1),x,k);
        b[i+(max-m)]:=x;
        end;
  {======суммирование==========}
  for i:=1 to max do
    sum[i]:= a[i]+b[i];
  for i:=max downto 2 do
    begin
        sum[i-1]:=sum[i-1] + sum[i] div 10;
        sum[i]:= sum[i] mod 10;
        end;
  {====вывод результата==========}
  for i:=1 to max do
    write (sum[i]);
end.
```

Тест:

Дано:	236754569081 937501
Результат:	236755506582

Задача Найти произведение двух сверхбольших чисел.

Идею решения иллюстрирует схема (рис. 3.3):

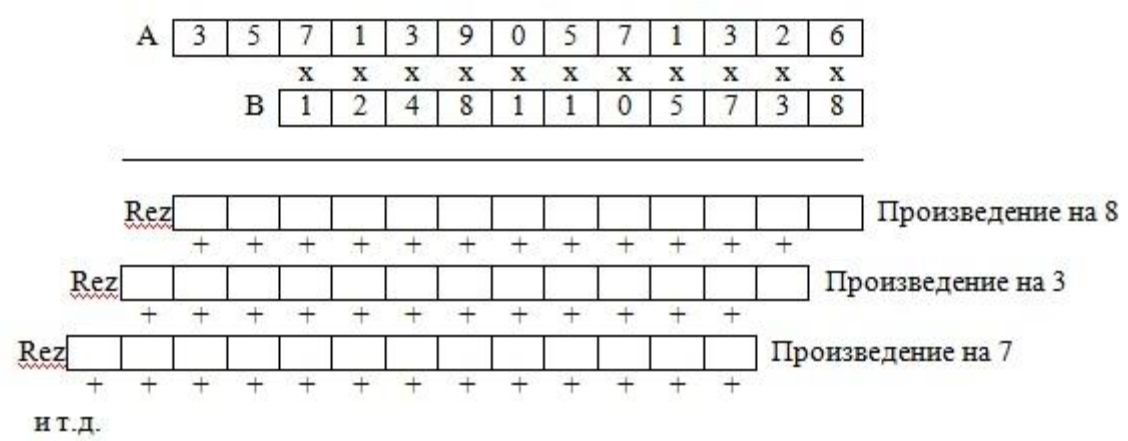

Рис. 3.3.

- возьмем последний элемент массива B; умножаем его на все элементы массива A. Результат храним в массиве Rez;
- суммируем два массива - Sum и сдвинутые на 1 позицию влево элементы массива Rez; Результат заносим в Sum;
- берем следующий элемент массива B (находящийся левее); повторяем с шага 2;
- выводим на экран содержимое массива Sum.

Программа на Бейсике:

```
input "первый сомножитель"; a$
input "второй сомножитель"; b$
n = len(a$)
m = len(b$)
dim a(n), b(m), rez(n), sum(n + m)
rem======разбор на цифры============
for i = 1 to n
  a(i) = val(mid$(a$, i, 1))
next
for i = 1 to m
  b(i) = val(mid$(b$, i, 1))
next
rem==произведение на j-ую цифру=========
for j = m to 1 step -1
  for i = 1 to n
    rez(i) = a(i) * b(j)
  next
  for i = n to 2 step -1
    rez(i - 1) = rez(i - 1) + rez(i) \ 10
        rez(i) = rez(i) mod 10
  next
  rem====сумма со сдвигом==========
  for i = 1 to n
    sum(i + j - 1) = sum(i + j - 1) + rez(i)
  next
  for i = n to 2 step -1
    sum(i - 1) = sum(i - 1) + sum(i) \ 10
        sum(i) = sum(i) mod 10
  next
next
rem====вывод результата==============
for i = 1 to n + m - 1
  print sum(i);
next
```

Программа на Паскале:

```
const mm=100;
var a,b, rez,sum: array [1..mm] of byte;
  i, j, n, m, max, x, k: integer;
  strokaA, strokaB: string;
begin
  writeln ( 'первый сомножитель'); readln (strokaA);
  writeln ( 'второй сомножитель'); readln (strokaB);
  n:= length(strokaA);
  m:= length(strokaB);
  {======разбор на цифры===============}
  for i:= 1 to n do
    begin
        val(copy(strokaA, i, 1),x,k);
        a[i]:= x;
        end;
  for i:= 1 to m do
    begin
        val(copy(strokaB, i, 1),x,k);
        b[i]:= x;
        end;
  {==произведение на j-ую цифру============}
  for j:= m downto 1 do
    begin
        for i:= 1 to n do
          rez[i]:= a[i] * b[j];
        for i:= n downto 2 do
          begin
          rez[i - 1]:= rez[i - 1] + rez[i] div 10;
          rez[i]:= rez[i] mod 10;
          end;
        {====сумма со сдвигом==============}
        for i:= 1 to n do
          sum[i + j - 1]:= sum[i + j - 1] + rez[i];
        for i:= n downto 2 do
          begin
          sum[i - 1]:= sum[i - 1] + sum[i] div 10;
          sum[i]:= sum[i] mod 10;
          end;
  end;
  {====вывод результата================}
  for i:= 1 to n + m - 1 do
    write (sum[i]);
end.
```

Тест:

Дано:	123456789 1234567
Результат:	1524156752330241363

Ключевые термины

- "Сверхбольшое число" - число, величина которого выходит за пределы диапазона допустимых значений выделенной для хранения числа ячейки памяти.

Краткие итоги

Операции со "сверхбольшими" числами выполняются по алгоритму:

- число вводится в строковую переменную.
- строка "разбирается на символы, каждый символ помещается в элемент массива, затем переводится в число.
- дальнейшие действия выполняются "поразрядно": умножение каждого элемента на другое число, организация переноса переполнения в старший разряд перенос.
- аналогично (поразрядно) выполняется операция суммирования двух "сверхбольших" чисел.

Набор для практики

Вопросы.

- Каким образом можно ввести с клавиатуры в переменную число, значение которого превышает 2147483647?
- Каким образом можно удвоить введенное с клавиатуры число, значение которого превышает 2147483647?

Упражнения.

- Вычислить N^K, (N и K>10).
- Вычислить N! (N-факториал) - произведение чисел натурального ряда до N включительно.

4. Лекция: Типовые алгоритмы обработки одномерных массивов. Сортировка методом "Пузырька"
вопросы | » для печати и PDA
Если Вы заметили ошибку - сообщите нам или выделите ее и нажмите Ctrl+Enter

В лекции продолжено знакомство с типовыми алгоритмами обработки одномерных массивов - рассмотрен типовой алгоритм сортировки элементов массива и разобраны некоторые олимпиадные задачи, которые решаются с использованием сортировки. **Цель лекции:** научиться применять изученный типовой алгоритм при решении классических задач.

Содержание

Сортировку данным методом рассмотрим на примере. Дан массив (табл.), элементы которого необходимо отсортировать в порядке возрастания:

5 8 4 9 3

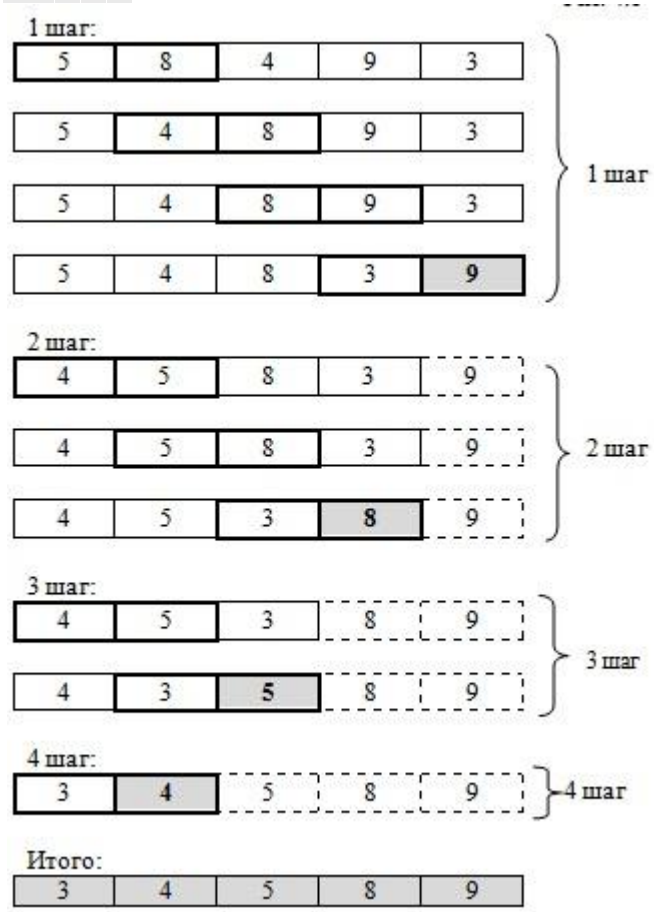

Рис. 4.1. Сортировка 'Пузырьком'

Программная реализация на Бейсике	Программная реализация на Паскале

`. . .`
`for j=n  to 2 step  -1`
`  for i=1  to j-1`
`    if   a(i)>a(i+1) then swap a(i), a(i+1)`
`next  i, j`
`…`
	`. . .`
	`for j:=n downto 2 do`
	`  for i:=1 to j-1 do`
	`    if   a[i]>a[i+1] then`
	`         begin`
	`         x:=a[i]; a[i]:= a[i+1]; a[i+1]:=x;`
	`         end;`
	`. . .`

Задачи, решаемые с использованием сортировки

Задача: Дан ряд целых случайных чисел. Преобразуйте его в ряд чисел, между которыми можно было бы расставить знаки "<" и ">" в чередующемся порядке.

Идея решения: Ряд чисел СОРТИРУЕМ. Меняем местами каждую пару чисел.

Решение задачи на Бейсике:

```
Input "количество чисел"; n
dim a(n)
for i = 1 to n
  input "введите число"; a(i)
next
rem=====сортируем массив==========
for j=n to 2 step -1
  for i=1 to j-1
        if a(i)>a(i+1) then swap a(i), a(i+1)
next i, j
rem ====меняем местами пару чисел=====
for i = 1 to n-1 step 2
  swap a(i), a(i+1)
next
print
for i=1 to n
  print a(i);
next
```

Решение задачи на Паскале:

```
var a: array [1..10] of integer;
  i,j,n,x:integer;
begin
  writeln ('количество чисел'); readln (n);
  for i:= 1 to n do
    begin
        writeln ('введите число'); readln (a[i]);
        end;
  {=====сортируем массив==========}
  for j:=n downto 2 do
    for i:=1 to j-1 do
          if a[i]>a[i+1] then
                begin
                x:=a[i];
                a[i]:= a[i+1];
                a[i+1]:=x;
```

```
              end;
  {====меняем местами пару чисел====}
  x:=1;
  for i:= 1 to (n div 2) do
    begin
        x:=a[j];
        a[j]:= a[j+1];
        a[j+1]:=x;
        j:=j+2;
        end;
  for i:=1 to n do writeln (a[i]);
end.
```

Тест:

Дано:	5, 2, 1, 8, 0, 67, 100.
Результат:	1, 0, 5, 2, 67, 8, 100

Задачи "Подпоследовательности"

1. В последовательности чисел выделить **все подпоследовательности подряд идущих чисел.**

   Идея решения: Ряд чисел СОРТИРУЕМ. Перебираем элементы массива, сравнивая два соседних элемента. Если разница между ними не равна единице, то начинаем печать новой подпоследовательности.

   Решение задачи на Бейсике:

```
Input "количество чисел"; n
dim a(n)
for i = 1 to n
  input "введите число"; a(i)
next
rem====сортируем массив==============
for j=n  to 2 step  -1
  for i=1  to  j-1
    if   a(i)>a(i+1) then swap a(i), a(i+1)
next  i, j
rem ============================
print a(1);
for i = 2 to n
  if a(i) - a (i-1) < > 1 then print
print a(i) ;
next
```

   Решение задачи на Паскале:

```
var      a: array [1..10] of integer;
  i,j,n,x: integer;
begin
  writeln ('количество чисел');
  readln (n);
  for i:= 1 to n do
    begin
  writeln ('введите число');
  readln (a[i]);
  end;
   {=====сортируем массив============}
```

```
  for j:=n  downto 2 do
    for i:=1  to  j-1 do
    if   a[i]>a[i+1] then
      begin
          x:=a[i];
          a[i:= a[i+1];
          a[i+1]:=x;
          end;
  {============================}
  write (a[1]);
  for i:= 2 to n do
         begin
    if (a[i]-a[i-1]) <> 1 then writeln;
  write (a[i]) ;
  end;
end.
```

Тест:

Дано:	N=10 2, 1, 5, 3, 7, 8, 6, 12, 9, 11
Результат:	1, 2, 3 5, 6, 7, 8, 9 11, 12

2. В последовательности чисел найти и вывести **подпоследовательность подряд идущих чисел наибольшей длины.**

   Идея решения: Ряд чисел СОРТИРУЕМ. Перебираем элементы массива, сравнивая два соседних элемента. Если разница между ними равна единице, значит очередное число входит в текущую подпоследовательность. В ячейке К накапливаем ее длину. Применив типовой алгоритм поиска МАКСИМАЛЬНОГО ЭЛЕМЕНТА находим подпоследовательность наибольшей длины.

   Решение задачи на Бейсике:

```
Input "количество чисел"; n
dim a(n)
for i = 1 to n
  input "введите число"; a(i)
next
rem====сортируем массив==============
for j = n to 2 step -1
  for i = 1 to j - 1
    if a(i) > a(i + 1) then swap a(i), a(i + 1)
next i, j
rem =выводим отсортированный массив======
for i = 1 to n
  print a(i);
next
print
max = 0
k = 1
for i = 1 to n - 1
  if a(i + 1) - a(i) = 1 then k = k + 1 else k = 1
```

```
  if k > max then max = k: number = i + 1
next i
print "максимальная длина="; max
print "самая длинная подпоследовательность="
for i=(number-max+1) to number
  print a(i);
next
```

Решение задачи на Паскале:

```
var a: array [1..10] of integer;
  max,k,number,i,j,n,x: integer;
begin
  writeln ('количество чисел');
  readln (n);
  for i:= 1 to n do
    begin
  writeln ('введите число'); readln (a[i]);
  end;
  {=====сортируем массив===============}
  for j:=n  downto 2 do
    for i:=1  to  j-1 do
    if   a[i]>a[i+1] then
      begin
          x:=a[i];
          a[i]:= a[i+1];
          a[i+1]:=x;
          end;
  {==============================}
  max:= 0;
  k:= 1;
  for i: = 1 to n-1 do
         begin
    if a[i+1] - a[i] = 1 then k: = k + 1
  else k: = 1;
  if k > max then
    begin
    max: = k;
    number: = i + 1;
     end;
    end;  writeln ( 'максимальная длина=', max);
  writeln ('самая длинная подпоследовательность=');
  for i: = (number - max + 1) to number do
  writeln (a[i]);
end.
```

Тест:

Дано:	N=10 2, 1, 5, 3, 7, 8, 6, 12, 9, 11
Результат:	Max=5 5, 6, 7, 8,9

Задача: Ввести предложение. Поменять порядок слов в нем, расположив их **в порядке увеличения букв** в словах.

Идея решения: Применив типовой алгоритм "РАЗБОРА" ПРЕДЛОЖЕНИЯ НА СЛОВА, помещаем каждое слово в элемент массива. Далее СОРТИРУЕМ массив слов, сравнивая количество букв в словах.

Решение задачи на Бейсике:

```
input "введите предложение"; a$
n=len (a$)
for i=1 to n
  if mid$(a$,i,1)=" " then k=k+1
next
k=k+1
dim a$(k)
j=1
for i=1 to n
       if mid$(a$,i,1)=" " then j=j+1 else a$(j)=a$(j)+mid$(a$,i,1)
next
rem=====сортируем массив======================
for j=k  to 2 step  -1
  for i=1  to  j-1
        if   len (a(i))>len (a(i+1)) then swap a(i), a(i+1)
next  i, j
rem ======================================
for i=1 to k
  print a(i); " ";
next
```

Решение задачи на Паскале:

```
var b:array[1..10] of string;
  a,a1,a2,x: string;
  n,i,j,k: integer;
begin
  writeln ('введите предложение');
  readln (a);
  n:=length (a);
  k:=0;
  for i:=1 to n do
    if copy(a,i,1)=' ' then k:=k+1;
  k:=k+1;
  j:=1 ;
  for i:=1 to n do
    if copy(a,i,1)=' ' then j:=j+1
       else b[j]:=b[j]+copy(a,i,1);
  {=====сортируем массив==========}
  for j:=k downto 2 do
    for i:=1 to j-1 do
          if length (b[i])>length (b[i+1]) then
            begin
                x:=b[i];
                b[i]:= b[i+1];
                b[i+1]:=x;
                end;
  { ============================}
  for i:=1 to k do
    writeln (b[i]);
end.
```

Тест:

Дано:	Мой любимый предмет - информатика

Результат: - Мой предмет любимый информатика

Задачи "Сортировка частей массива"

Среди задач на сортировку элементов массива можно выделить большую группу задач, в которых необходимо отсортировать ЧАСТЬ МАССИВА ПО ВОЗРАСТАНИЮ, ЧАСТЬ ПО УБЫВАНИЮ.

Идея решения: Допустим, дан массив A, заполненный положительными и отрицательными целыми числами (рис. 4.2):

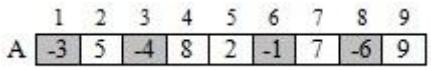

Рис. 4.2.

Необходимо отсортировать положительные элементы массива по возрастанию, отрицательные элементы оставить на своих местах.

- Вводим дополнительный массив B, который заполняем индексами положительных элементов массива A (рис. 4.3):

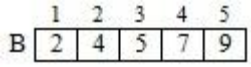

Рис. 4.3.

- Затем используем ТИПОВОЙ АЛГОРИТМ СОРТИРОВКИ, в качестве индексов перебираемых в цикле элементов массива A используем элементы массива B. Данное решение задачи также можно выделить в отдельный типовой алгоритм:

  Программная реализация на Бейсике:

```
Input "количество чисел"; n
dim a(n), b(n)
for i = 1 to n
  input "введите число"; a(i)
next
j=1
for i=1 to n
  if a(i)>0 then b(j)=i: j=j+1:k=k+1
next i
for j=k to 2 step -1
  for i=1 to j-1
  if a(b(i))>a(b(i+1)) then swap a(b(i)),a(b(i+1))
next i,j
for i=1 to n
  print a(i); "  ";
next i
```

  Программная реализация на Паскале:

```
const m=10;
var      a, b: array [1..m] of integer;
```

```
  i, j, n, x, k: integer;
begin
  writeln ('количество чисел');
  readln (n);
  for i:= 1 to n do
    begin
  writeln ('введите число');
  readln (a[i]);
  end;
  j:=1;
  k:=0;
  for i:=1 to n do
  if a[i]>0 then
    begin
    b[j]:=i;
    j:=j+1;
    k:=k+1;
    end;
  for j:=k downto 2 do
     for i:=1 to j-1 do
    if a[b[i]]>a[b[i+1]] then
    begin
    x:=a[b[i]];
    a[b[i]]:=a[b[i+1]];
    a[b[i+1]]:=x;
    end;
  for i:=1 to n do
    write (a[i],' ');
end.
```

Тест:

Дано:	N=9 -3, 5, -1, 3, 6, -8, 2, -1, -3
Результат:	-3, 2, -1, 3, 5, -8, 6, -1, -3

Ключевые термины

- Сортировка пузырьком - один из методов упорядочивания элементов одномерного массива.
- Подпоследовательность - фрагмент последовательности данных, выделенный по определенному признаку.

Краткие итоги

Сортировка элементов одномерного массива методом "Пузырька" основывается на сравнении двух соседних элементов. Элементы меняются местами, если больший элемент (при сортировке по возрастанию) оказывается левее меньшего элемента. За один проход элементов массива наибольший элемент оказывается в конце массива. Далее алгоритм повторяется с первого элемента до предпоследнего и т.д.

При сортировке частей массива (в зависимости от критериев отбора элементов в выбираемой части массива) в качестве индексов перебираемых в цикле элементов исходного массива A используем элементы дополнительного массива B (в нем хранятся индексы выбираемых элементов исходного массива).

Набор для практики

Вопросы.

- Что происходит за один проход массива при сортировке "Пузырьком"?
- Объясните зависимость счетчика внутреннего цикла от счетчика внешнего цикла в типовом алгоритме сортировки одномерного массива "Пузырьком".

Упражнения.

- Символьная строка состоит из цифр и латинских букв. Упорядочьте все цифры в порядке убывания, а буквы в алфавитном порядке, оставляя и те и другие на своих местах.
- В массиве натуральных чисел все простые числа упорядочьте по возрастанию, составные по убыванию.
- Найдите в тексте все гласные буквы и расположите их в порядке возрастания номеров алфавита, а все согласные буквы - в порядке убывания номеров алфавита.
- Ввести предложение. Измените порядок слов в предложении, расположив в алфавитном порядке (по первой букве каждого слова).

В лекции рассматриваются типовые алгоритмы обработки двумерных массивов. рассмотрены некоторые олимпиадные задачи, которые решаются с использованием этих алгоритмов. **Цель лекции:** научиться применять изученные типовые алгоритмы при решении классических задач.

Содержание

Обработку двумерного массива можно условно разделить на три группы:

- Обработка всего массива;
- Обработка отдельно по строкам и столбцам;
- Обработка относительно диагоналей.

Рассмотрим типовые алгоритмы обработки двумерных массивов каждой группы в отдельности.

Обработка всего массива

Таблица 5.1.

Типовой алгоритм	Программная реализация (Бейсик)	Программная реализация (Паскаль)

Заполнение (порядок обхода элементов двумерного массива - построчно слева направо)	…
	for i=1 to n
	for j=1 to m
	input x(i,j)
	next j
	next i
	…
		…
		for i:=1 to n do
		for j:=1 to m do
		readln (x[i,j]);
		…

Вывод	…
	for i=1 to n
	for j=1 to m
	print x(i,j);" ";
	next j
	print
	next i
		…
		for i:=1 to n do
		begin
		for j:=1 to m do
		write (x[i,j], ' ');
		writeln;
		end;
		…

Сумма, произведение	…
	p=1
	for i=1 to n
		…
		s:=0; p:=1;
		for i:=1 to n do

	for j=1 to m
	s=s+x(i,j)
	p=p*x(i,j)
	next j
	next i
	…
		for j:=1 to m do
		begin
		s:=s+x[i,j];
		p:=p*x[i,j];
		end
		…

Максимальный (минимальный) элемент	…
	max=x(1,1)
	min=x(1,1)
	for i=1 to n
	for j=1 to m
	if x(i,j)>max then
	max=x(i,j)
	if x(i,j)<min then
	min=x(i,j)
	next j,i
		…
		max:=x[1,1];
		min:=x[1,1];
		for i:=1 to n do
		for j:=1 to m do
		begin
		if x[i,j]>max then
		max:=x[i,j];
		if x[i,j]<min then
		min:=x[i,j];
		end
		…

Выбор по условию	…
	for i=1 to n
	for j=1 to m
	if {условие} then
	{действие}
	next j
	next i
	…
		…
		for i:=1 to n do
		for j:=1 to m do
		if {условие} then
		{действие}
		…

Типовые алгоритмы обработки двумерного массива отдельно по строкам:

Таблица 5.2.

Типовой алгоритм	Программная реализация (Бейсик)	Программная реализация (Паскаль)

Сумма элементов каждой строки	…
	for i=1 to n
	for j=1 to m
	s(i)=s(i)+x(i,j)
	next j
	next i
	rem----------
	for i=1 to n
	print s(i);" ";
	next
		…
		for i:=1 to n do
		s[i]:=0;
		for i:=1 to n do
		for j:=1 to m do
		s[i]:=s[i]+x[i,j];
		for i:=1 to n do
		write (s[i]);
		…

Произведение элементов каждой строки	…
	for i=1 to n
	p(i)=1
	next
	rem----------
	for i=1 to n
	for j=1 to m
	p(i)=p(i)*x(i,j)
	next j
	next i
	rem----------
	for i=1 to n
	print p(i);" ";
	next
		…
		for i:=1 to n do
		p[i]:=1;
		for i:=1 to n do
		for j:=1 to m do
		p[i]:=p[i]*x[i,j];
		for i:=1 to n do
		write (p[i]);

Максимальный (минимальный) элемент каждой	…
	for i= 1 to n
	max(i)=x(i,1)
	min(i)=x(i,1)
		…
		for i:= 1 to n do
		begin
		max[i]:=x[i,1];

строки	next
	rem----------
	for i=1 to n
	for j=1 to m
	if x(i,j)>max(i) then max(i)=x(i,j)
	if x(i,j)<min(i) then min(i)=x(i,j)
	next j,i
	rem----------
	for i=1 to n
	print max(i);" ";;
	next
	print
	for i=1 to n
	print min(i);" ";;
	next
	…
		min[i]:=x[i,1];
		end;
		for i:=1 to n do
		for j:=1 to m do
		begin
		if x[i,j]>max[i]
		then max[i]:=x[i,j];
		if x[i,j]<min[i]
		then min[i]:=x[i,j];
		end;
		for i:=1 to n do
		write (max[i]);
		writeln;
		for i:=1 to n do
		write (min[i]);
		…

Выбор по условию (в каждой строке)	…
	for i=1 to n
	for j=1 to m
	if {условие} then {rez(i)=…}
	next j
	next i
	rem----------
	for i=1 to n
	print rez(i);" ";
	next
		…
		for i:= 1 to n do
		begin
		rez[i]:=0;
		end;
		for i:=1 to n do
		for j:=1 to m do
		if {условие} then
		{rez[i]:=…};
		for i:=1 to n do
		write (rez[i]);
		…

Типовые алгоритмы обработки двумерного массива отдельно по столбцам:

Таблица 5.3.

Типовой алгоритм	Программная реализация (Бейсик)	Программная реализация (Паскаль)

Сумма элементов в каждом столбце	…
	for j=1 to m
	for i=1 to n
	s(j)=s(j)+x(i,j)
	next i
	next j
	rem----------
	for j=1 to m
	print s(j)
	next
		…
		for j:=1 to m do
		s[j]:=0;
		for j:=1 to m do
		for i:=1 to n do
		s[j]:=s[j]+x[i,j];
		for j:=1 to m do
		write (s[j]);
		…

Произведение элементов в каждом столбце	…
	for j=1 to m
	p(j)=1
	next
	rem----------
	for j=1 to m
	for i=1 to n
	p(j)=p(j)*x(i,j)
	next i
	next j
	rem----------
	for j=1 to m
	print p(j)
	next
		…
		for j:=1 to m do
		p[j]:=1;
		for j:=1 to m do
		for i:=1 to n do
		p[j]:=p[j]*x[i,j];
		for j:=1 to m do
		write (p[j]);
		…

Максимальный (минимальный)	… for j= 1 to m	… for j:= 1 to m do

элемент в каждом столбце	max(j)=x(i,1)
	next
	rem----------
	for j=1 to m
	for i=1 to n
	if x(i,j)>max(j) then max(j)=x(i,j)
	next i,j
	rem----------
	for j=1 to m
	print max(j);" ";
	next
	print
	for j=1 to m
	print min(j);" ";
	next
		begin
		max[j]:=x[i,1];
		min[j]:=x[i,1];
		end;
		for j:=1 to m do
		for i:=1 to n do
		begin
		if x[i,j]>max[j]
		then max[j]:=x[i,j]
		if x[i,j]<min[j]
		then min[j]:=x[i,j]
		end;
		for j:=1 to m do
		write (max[j]);
		writeln;
		for j:=1 to m do
		write (min[j]);
		…

Выбор по условию (в каждом столбце)	…
	for j=1 to m
	for i=1 to n
	if {усл.} then {rez(i)=…}
	next i
	next j
	rem----------
	for j=1 to m
	print rez(j);" ";
	next
		…
		for j:= 1 to m do
		rez[j]:=0;
		for j:=1 to m do
		for i:=1 to n do
		if {усл.} then
		{rez[i]:=…};
		for j:=1 to m do
		write (rez[j]);
		…

Типовые алгоритмы обработки двумерного массива относительно диагоналей

Если количество строк и столбцов в двумерном массиве одинаково, то такой массив называется квадратным. Типовые алгоритмы для квадратных массивов позволяют обрабатывать массив относительно его диагональных элементов. Зависимость индексов элементов, расположенных на побочной диагонали легко определяется при после повторения Лекции 1 (номер строки элемента растет, номер столбца убывает).

Рис. 5.1.

Главная диагональ. В таблице приведены типовые алгоритмы обработка элементов двумерного массива, расположенных НА, ВЫШЕ и НИЖЕ главной диагонали.

Таблица 5.4.

Типовой алгоритм	Программная реализация (Бейсик)	Программная реализация (Паскаль)

Сумма элементов, расположенных НА главной диагонали	…
	for i=1 to n
	s=s+x(i,i)
		…
		s:=0;
		for i:=1 to n do

	next i …	s:=s+x[i,i]; …

Сумма элементов, расположенных ВЫШЕ главной диагонали	…
	for i=1 to n
	for j=1 to n
	if i<j then
	s=s+x(i,j)
	next j, i
	…
		…
		s:=0;
		for i:=1 to n do
		for j:=1 to n do
		if i<j then
		s:=s+x[i,j];
		…

Сумма элементов, расположенных НИЖЕ главной диагонали	…
	for i=1 to n
	for j=1 to n
	if i>j then
	s=s+x(i,j)
	next j i
	…
		…
		s:=0;
		for i:=1 to n do
		for j:=1 to n do
		if i>j then
		s:=s+x[i,j];
		…

Побочная диагональ. В таблице приведены типовые алгоритмы обработка элементов двумерного массива, расположенных НА, ВЫШЕ и НИЖЕ побочной диагонали.

Таблица 5.5.

Типовой алгоритм	Программная реализация (Бейсик)	Программная реализация (Паскаль)

Сумма элементов, расположенных НА побочной диагонали	…
	for i=1 to n
	s=s+x(i, n-i+1)
	next i
	…
		…
		s:=0;
		for i:=1 to n do
		s:=s+x[i, n-i+1];
		…

Сумма элементов, расположенных ВЫШЕ побочной диагонали	…
	for i=1 to n
	for j=1 to n
	if (i<n-j+1) then
	s=s+x(i,j)
	next j, i
	…
		…
		s:=0;
		for i:=1 to n do
		for j:=1 to n do
		if (i<n-j+1) then
		s:=s+x[i,j];
		…

Сумма элементов, расположенных НИЖЕ побочной диагонали	…
	for i=1 to n
	for j=1 to n
	if (i>n-j+1) then
	s=s+x(i,j)
	next j, i
	…
		…
		s:=0;
		for i:=1 to n do
		for j:=1 to n do
		if (i>n-j+1) then
		s:=s+x[i,j];
		…

Обработка квадратной матрицы относительно диагоналей (рациональный обход)

Предложенные выше типовые алгоритмы обработки квадратного массива относительно диагоналей нерациональны, т.к. перебираются все элементы массива. Более рационально ограничить перебор элементов. Решит проблему **введение зависимости начального или конечного значений управляющей переменной внутреннего цикла от значения счетчика внешнего цикла.**

Рассмотрим типовые алгоритмы обработки квадратного массива относительно диагоналей рациональным способом на примерах.

Задача: заполнить элементы квадратного массива "1" так, как показано на рисунке:

Таблица 5.6.

Ниже и на главной диагонали	Выше и на главной диагонали	Выше и на побочной диагонали	Ниже и на побочной диагонали
100000000 **11**0000000 **111**000000 **1111**00000 **11111**0000 **111111**000 **1111111**00 **11111110** **111111111**	**111111111** 0**11111111** 00**1111111** 000**111111** 0000**11111** 00000**1111** 000000**111** 0000000**11** 00000000**1**	**111111111** **1111111**0 **111111**00 **11111**000 **1111**0000 **111**00000 **11**000000 **1**0000000 **100000000**	000000000**1** 000000**11** 00000**111** 0000**1111** 000**11111** 00**111111** 0**1111111** **111111111**

Программная реализация на Бейсике:
 	
…
for i=1 to n
for j=1 to i
x(i,j)=1
next j, i
…
 	
Программная реализация на Паскале:
 	
…
for i:=1 to n do
for j:=1 to i do
x[i,j]:=1;
…
	Программная реализация на Бейсике:
	
	…
	for i=1 to n
	for j=i to n
	x(i,j)=1
	next j, i
	…
	
	Программная реализация на Паскале:
	
	…
	for i:=1 to n do
	for j:=i to n do
	x[i,j]:=1;
	…
		Программная реализация на Бейсике:
		
		…
		for i=1 to n
		for j=1 to (n-i+1)
		x(i,j)=1
		next j, i
		…
		
		Программная реализация на Паскале:
		
		…
		for i:=1 to n do
		for j:=1 to (n-i+1) do
		x[i,j]:=1;
		…
			Программная реализация на Бейсике:
			
			…
			for i=1 to n
			for j=(n-i+1) to n
			x(i,j)=1
			next j, i
			…
			
			Программная реализация на Паскале:
			
			…
			for i:=1 to n do
			for j:=(n-i+1) to n do
			x[i,j]:=1;
			…

Ключевые термины

- Двумерный массив -именованный набор однотипных переменных, расположенных в памяти непосредственно друг за другом, доступ к которым осуществляется по индексу. Массивы с одним индексом называют одномерными, с двумя - двумерными.
- Квадратный массив - двумерный массив, количество строк и столбцов в котором одинаково.

Краткие итоги

Для решения задач с использованием двумерных массивов необходимо воспользоваться типовыми алгоритмами обработки, такими как:

- Обработка всего массива:
  - Заполнение, вывод
  - Сумма, произведение
  - Максимальный (минимальный) элемент
  - Выбор по условию
- Обработка отдельно по строкам и столбцам:
  - Заполнение, вывод
  - Сумма, произведение
  - Максимальный (минимальный) элемент
  - Выбор по условию
- Обработка относительно диагоналей:
  - Заполнение, вывод
  - Сумма, произведение
  - Максимальный (минимальный) элемент
  - Выбор по условию

При обработке квадратного массива относительно его диагоналей необходимо воспользоваться рациональным способом, исключающим обход всех элементов массива.

Набор для практики

Вопросы.

- Что произойдет если поменять местами счетчики внешнего и внутреннего цикла i и j в заголовках циклов (в типовом алгоритме обработки двумерного массива)?
- Какова зависимость индексов элементов, расположенных на главной диагонали квадратного массива? Побочной?

Упражнения.

- Найдите максимальный элемент из минимальных элементов каждой строки двумерного массива и минимальный элемент из максимальных элементов каждого столбца двумерного массива размерностью NxM.
- Заполните квадратные массивы, как предложено на рис. 5.2:

Задача 1	*Задача 2*	*Задача 3*	*Задача 4*
100000000	000000001	111111111	000000000
110000000	000000011	011111110	000000000
111000000	000000111	001111100	000000000
111100000	000001111	000111000	000000000
111110000	000011111	000010000	000010000
111100000	000001111	000000000	000111000
111000000	000000111	000000000	001111100
110000000	000000011	000000000	011111110
100000000	000000001	000000000	111111111

Рис. 5.2.

6. Лекция: Задачи, сгруппированные по методам решения. Использование дополнительного массива "флажков"
для печати и PDA
Если Вы заметили ошибку - сообщите нам или выделите ее и нажмите Ctrl+Enter

Что еще, кроме подбора типовых алгоритмов поможет в подготовке к олимпиадам? Огромную помощь может оказать группировка задач - не по темам (задачи на строки, задачи из теории чисел и др.), а по методам решения этих задач. Рассмотрим некоторые методы решения часто встречающихся в практике программирования задач. В данной лекции будет рассмотрен прием "Использование флажков", отработаны типовые алгоритмы обработки одномерных массивов. **Цель лекции:** научиться применять изученный метод при решении классических задач.

Содержание

Для решения некоторых задач на одномерные массивы часто бывает необходимо каким-либо образом отметить элементы, удовлетворяющие условию. Для этого резервируют **дополнительный массив** Flag, элементы которого заполняют единицами, если соответствующие элементы исходного массива удовлетворяет условию.

Если условий несколько, то "флажки" принимают разное значение (например: "1" и "-1" и др.). Рассмотрим данный прием на практике.

Задача 1: Написать программу для поиска всех простых чисел (до числа N).

Дополнительные сведения: Рассмотрим алгоритм поиска простых чисел на диапазоне от 1 до N методом "**Решето Эратосфена**". Проиллюстрируем этот метод:

- 1 шаг: берем "2", отмечаем серым цветом. Вычеркиваем все числа, кратные двум.
- 2 шаг: берем следующее за "2" невычеркнутое число ("3"), отмечаем серым цветом. Вычеркиваем все числа, кратные трем и т.д.
- В "решете" остались числа, отмеченные серым цветом. Это простые числа.

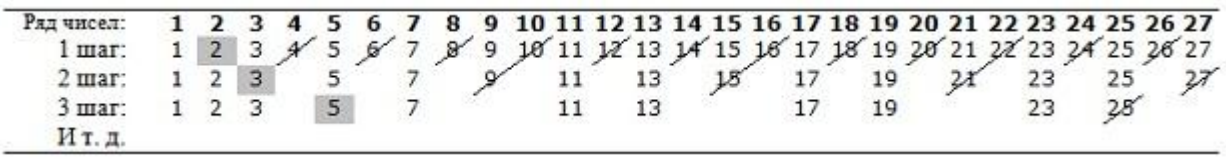

Рис. 6.1.

Идея решения:

- Заполняем одномерный массив A подряд идущими числами до N (например: N=22) включительно (таблица ниже):

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22

- Заполняем элементы массива Flag флажками-единицами, если соответствующие элементы массива A кратны "2" (рис. 6.2):

A	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
flag				1		1		1		1		1		1		1		1		1		1

Рис. 6.2.

- Затем заполняем элементы массива Flag флажками-единицами, если соответствующие элементы массива A кратны "3" (рис. 6.3):

A	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
flag				1		1		1	1	1		1		1	1	1		1		1	1	1

Рис. 6.3.

- и т.д.
- В "решете" (массиве A) остались числа, отмеченные серым цветом (рис. 6.4):

A	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
flag				1		1		1	1	1		1		1	1	1		1		1	1	1

Рис. 6.4.

Для сокращения количества шагов достаточно перебирать элементы массива A до половины (т.к. максимальный делитель n - это n/2).

Решение задачи на Бейсике:

```
input "n="; n
dim a(n), flag(n)
for i=1 to n
  a(i)=i
next
for i = 2 to n/2
  if flag(i)=0 then
   for j=i+1 to n
    if a(j) mod a(i)=0 then flag(j)=1
   next j
  end if
next i
for i=1 to n
  if flag(i)=0 then print a(i);
next
```

Решение задачи на Паскале:

```
Program pr;
Var a,flag:array [1..100] of integer;
```

```
  I,j,n:integer;
begin
  writeln ('n=');
  readln (n);
  for i:=1 to n do
   a[i]:=i;
  for i:=2 to n div 2 do
   if flag[i]=0 then
    for j:=i+1 to n do
         if (a[j] mod a[i]=0) then flag[j]:=1;
  for i:=1 to n do
     if flag[i]=0 then writeln (a[i]);
end.
```

Тест:

Дано:	100
Результат:	1 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 97

Задача 2: В одномерном массиве, заполненном целыми числами подсчитать число различных элементов.

Идея решения: сравнивая очередной элемент массива A с остальными элементами, заполняем единицей соответствующий повторяющемуся элементу элемент массива Flag. Количество нулевых элементов в массиве Flag и будет равно количеству различных элементов массива A.

Решение задачи На Бейсике:

```
input "количество чисел"; n
dim a(n), flag(n)
for i = 1 to n
  input "введите число"; a(i)
next
for i = 1 to n - 1
  if flag(i) = 0 then
    for j = i + 1 to n
          if a(i)=a(j) then flag(j)=1
        next
  end if
next
for i = 1 to n
  if flag(i)=0 then k = k+1
next
print "k="; k
```

Решение задачи на Паскале:

```
Program pr;
var     a,flag: array [1..100] of integer;
  i,j,n,k: integer;
begin
  writeln ('количество чисел');
  readln (n);
  for i:= 1 to n do
    begin
        writeln ('введите число');
        readln (a[i]);
        end;
```

```
  {=========================}
  for i:=1 to n-1 do
        if flag[i]=0 then
          for j:=i+1 to n do
            if a[i]=a[j] then flag[j]:=1;
  for i:=1 to n do
        if flag[i]=0 then k:=k+1;
  writeln ('k=', k);
end.
```

Тест:

Дано:	n=10 1, 4, 2, 2, 3, 1, 2, 3, 4, 1
Результат:	k=4

Задача 3: Вывести элемент, встречающийся в одномерном массиве чаще других.

Идея решения: При сравнении элементов массива А находим повторяющиеся элементы. Массив Flag заполняем количеством повторений элемента массива А. Затем, применив типовой алгоритм ПОИСКА МАКСИМАЛЬНОГО ЭЛЕМЕТА МАССИВА находим позицию элемента массива А, встречающегося чаще других.

Решение задачи на Бейсике:

```
input "количество чисел";n
dim a(n), flag(n)
for i = 1 to n
  input "введите число"; a(i)
next
for i = 1 to n - 1
  if flag(i) = 0 then
        for j = i + 1 to n
          if a(i) = a(j) then k = k + 1: flag(j) = k
    next
        k = 0
  end if
next
for i = 1 to n
        if flag(i) > max then max = flag(i): b = i
next
print "чаще встречается "; a(b);
```

Решение задачи на Паскале:

```
var a,flag: array [1..100] of integer;
  i,j,n,k,max,b: integer;
begin
  writeln ('количество чисел');
  readln (n);
  for i:= 1 to n do
    begin
        writeln ('введите число');
        readln (a[i]);
        end;
  {==================}
  k:=0;
  for i:=1 to n-1 do
        if flag[i]= 0 then
```

```
          for j:=i+1 to n do
                begin
                if a[i]= a[j] then
                  begin
                  k:= k+1;
                  flag[j]:=k;
                  end;
                k:=0;
                end;
  max:=flag[1];
  for i:=1 to n do
        if flag[i]>max then
          begin
          max:=flag[i];
          b:=i;
          end;
  writeln ('чаще встречается ', a[b]);
end.
```

Тест:

Дано:	`n=10` `1, 4, 2, 2, 3, 1, 2, 3, 2, 1`
Результат:	Чаще встречается 2

Ключевые термины

- Метод "Решето Эратосфена" - алгоритм поиска простых чисел на диапазоне от 1 до N.

Краткие итоги

Для решения некоторых задач на одномерные массивы часто бывает необходимо каким-либо образом отметить элементы, удовлетворяющие условию. Для этого резервируют дополнительный массив флажков, элементы которого заполняют единицами, если соответствующие элементы исходного массива удовлетворяет условию.

В алгоритме "Решето Эратосфена" флажками отмечаются элементы, которые необходимо исключить из дальнейшего рассмотрения ("вычеркнуть").

При необходимости флажки могут принимать не только единичные значения - они могут "указывать" на разнообразные состояния элементов рассматриваемого массива.

Набор для практики

Вопросы.

- Продолжите фразу: "Для того, чтобы отметить элементы, удовлетворяющие условию, используют…".
- В чем заключается алгоритм поиска простых чисел на диапазоне от 1 до N методом "Решето Эратосфена"?

Упражнения.

- В одномерном массиве, заполненном целыми числами подсчитать количество повторяющихся элементов.
- Ввести предложение, разобрать его на слова, поместив каждое слово в ячейку массива. Подсчитать, сколько слов в предложении имеют одинаковое количество букв.

7. Лекция: Задачи, сгруппированные по методам решения. Использование дополнительного массива "флажков" (три задачи - один алгоритм)

Продолжаем отрабатывать пройденный прием (использование дополнительного массива "флажков") в ситуациях, когда необходимо отметить наступление и окончание какого-то события (открытие-закрытие скобок, приход-уход сторожа, начало-конец отрезка). При наступлении события "флажок" примет значение "1", при окончании - "-1". Материалы лекции базируются на типовых алгоритмах обработки одномерных массивов и строк.
Цель лекции: научиться применять изученный метод при решении классических задач.

Содержание

Рассмотрим несколько практических задач.

Задача 1: В строке, содержащей арифметическое выражение проверить, правильно ли расставлены скобки. В случае лишних скобок - не считать неправильным расстановку скобок (например: ((а+в)) -скобки расставлены верно).

Идея решения:

1. Допустим, дано арифметическое выражение: (а + в ((4а - 8) в + 3) - 15в)
2. Резервируем два массива: A$ и Flag. "Разбираем" строку, в которой хранится арифметическое выражение (применив типовой алгоритм "РАЗБОРА" СТРОКИ НА СИМВОЛЫ), помещая каждый символ в элемент массива A$.

   Массив Flag заполняем флажками:

   - "1" - если элемент массива A$ - открывающаяся скобка;
   - "-1" - если элемент массива A$ - закрывающаяся скобка (рис. 7.1).

A$	(	а	+	в	(	(	4	а	-	8	)	в	+	3	)	-	1	5	в	)
Flag	1	0	0	0	1	1	0	0	0	0	-1	0	0	0	-1	0	0	0	0	-1

Рис. 7.1.

3. Суммируем элементы массива Flag. Если в процессе перебора элементов сумма станет отрицательной, значит скобки расставлены неверно. Итоговая сумма должна быть равна 0.

   Решение задачи на Бейсике:

```
input "введите ар. выражение"; stroka$
n=len(stroka$)
dim a$ (n), flag (n)
rem-
for i=1 to n
 a$(i)=mid$(stroka$, i, 1)
next
rem======================
for i=1 to n
 if a$(i)='(' then flag (i)=1
 if a$(i)=')' then flag (i)=-1
next
rem======================
for i=1 to n
 s=s+flag (i)
 if s<0 then x=1
next
if s=0 and x=0 then ?"скобки расставлены верно" else ?"скобки
расставлены неверно"
```

Решение задачи на Паскале:

```
const    m=10;
var flag: array [1..m] of integer;
 a: array [1..m] of string[1];
 stroka: string;
 i,s,n,x: integer;
begin
 writeln ('введите ар. выражение'); readln (stroka);
 n:=length (stroka);
 for i:=1 to n do
  begin
  a[i]:=copy(stroka, i, 1);
  flag[i]:=0;
  end;
 for i:=1 to n do
  begin
  if a[i]="(" then flag [i]:=1;
  if a[i]=")" then flag [i]:=-1;
  end;
 s:=0;
 for i:=1 to n do
  begin
  s:=s+flag [i];
  if s<0 then x:=1;
  end;
 if (s=0) and (x=0) then writeln ('скобки расставлены верно')
 else  writeln ('скобки расставлены неверно');
end.
```

Тест:

Дано:	(4*5+1)*((2/6-2*3)+4))	(6+56))-90*(5-2*(3-7/3)
Результат:	скобки расставлены верно	скобки расставлены неверно

Задача 2: В картинной галерее работают сторожа. Для каждого сторожа известно время прихода на работу и время ухода. Определить, всегда ли галерея охраняется.

Идея решения:Пример (табл. 7.1):

Таблица 7.1.		
	Время прихода	**Время ухода**
1 сторож	8.00	12.00
2 сторож	11.00	16.00
3 сторож	15.00	19.30
4 сторож	20.00	23.50

1. Заполняем массивы:

   A: временем прихода и ухода сторожей;

   Flag:

   - "1" - если соответствующий элемент массива A - время прихода сторожа;
   - "-1" - если элемент массива A - время ухода сторожа;

A	8.00	12.00	11.00	16.00	15.00	19.30	20.00	23.50
Flag	1	-1	1	-1	1	-1	1	-1

Рис. 7.2.

2. СОРТИРУЕМ массив A, одновременно переставляя элементы массива flag:

A	8.00	11.00	12.00	15.00	16.00	19.30	20.00	23.50
Flag	1	1	-1	1	-1	-1	1	-1

Рис. 7.3.

3. Суммируем элементы второго массива. Если текущая сумма обнулилась (но конец массива не достигнут), то, значит, галерея осталась без охраны (в 19.30 сторож ушел, а смена еще не пришла).

A	8.00	11.00	12.00	15.00	16.00	19.30	20.00	23.50
Flag	1	1	-1	1	-1	-1	1	-1
	Сумма=1	Сумма=2	Сумма=1	Сумма=2	Сумма=1	**Сумма=0**	Сумма=1	Сумма=0

Рис. 7.4.

Решение на Бейсике:

```
input "количество сторожей="; n
dim a(2*n), flag(2*n)
for i=1 to n*2 step 2
 input "время прихода="; a(i)
 flag (i) = 1
 input "время ухода="; a(i + 1)
 flag (i + 1) = -1
```

```
next
rem==сортировка=====================================
for j = 2*n to 2 step -1
 for i = 1 to j - 1
  if a(i) > a(i + 1) then swap a(i), a(i + 1): swap flag(i), flag(i +
1)
next i, j
rem=============================================
k = 0
for i = 1 to 2*n
 s = s + flag(i)
 if s = 0 then k = k + 1
next
if k = 1 then print "охранялась всегда" else print "не охранялась "; k
- 1; " раз"
```

Решение задачи на Паскале:

```
const m=20;
var a, flag: array [1..m] of integer;
 i,s,k,n,x: integer;
begin
 writeln ('количество сторожей');
 readln (n);
 j:=1;
 for i:=1 to n do
  begin
  writeln ('время прихода, ухода');
  readln (a[j], a[j+1]);
  flag [j]:=1;
  flag [j+1]:=-1;
  j:=j+2;
  end;
 for j:=2*n downto 2 do
  for i:=1 to j-1 do
   if a[i]>a[i+1] then
 begin
 x:=a[i];
 a[i]:=a[i+1];
 a[i+1]:=x;
 x:=flag[i];
 flag [i]:=flag [i+1];
 flag [i+1]:=x;
 end;
 k:=0;
 s:=0;
 for i:=1 to 2*n do
  begin
  s:=s+flag [i];
  if s=0 then k:=k+1;
  end;
 if k=1 then writeln ('галерея всегда охранялась')
         else writeln ('галерея оставалась без охраны',k-1,'раз');
end.
```

Тест:

Дано:	n=3 8 10 9 12	n=3 8 10 9 12

	11 13	13 14
Результат:	охранялась	не охранялась 1 раз

Задача 3: N отрезков на координатной прямой заданы координатами своих концов. Определить количество **связных областей**.

Идея решения:

1. Заполняем массивы:

   A: координатами начала и конца отрезков;

   Flag:

   - "1" - если соответствующий элемент массива A - координата начала отрезка;
   - "-1" - если элемент массива A - координата конца отрезка;
2. СОРТИРУЕМ массив A, одновременно переставляя элементы массива Flag.
3. Суммируем элементы массива Flag. Если текущая сумма обнулилась (но конец массива не достигнут), то, значит, получена одна связная область. Количество обнулений и будет количеством связных областей.

   Решение на Бейсике:

```
input "введите количество отрезков", n
dim a (2*n), flag (2*n)
for i=1 to n*2 step 2
 input "первая координата", a(i)
 flag (i)=1
 input "вторая координата", a(i+1)
 flag (i+1)=-1
next
for j=2*n to 2 step -1
 for i= 1 to j-1
  if a(i)>a(i+1) then swap a(i), a(i+1): swap flag(i), flag(i+1)
next i, j
for i=1 to 2*n
 s=s+flag (i)
 if s=0 then k=k+1
next
print "количество связных областей ="; k
```

   Решение задачи на Паскале:

```
const m=20;
var a, flag: array [1..m] of integer;
 i,j,s,n,k,x: integer;
begin
 writeln ('количество отрезков'); readln (n);
 j:=1;
 for i:=1 to n do
  begin
  writeln ('введите координаты начала и конца отрезка');
  readln (a[j], a[j+1]);
  flag [j]:=1;
  flag [j+1]:=-1;
  j:=j+2;
```

```
 end;
for j:=2*n downto 2 do
 for i:=1  to j-1 do
  if a[i]>a[i+1] then
 begin
  x:=a[i];
  a[i]:=a[i+1];
  a[i+1]:=x;
  x:=flag[i];
  flag [i]:= flag [i+1];
  flag [i+1]:=x;
  end;
s:=0;
k:=0;
for i:=1 to 2*n do
 begin
 s:=s+flag [i];
 if s=0 then k:=k+1;
 end;
writeln ('количество связных областей=', k);
end.
```

Тест:

Дано:	n=3 1 5 4 9 11 15	n=3 1 3 4 9 11 15
Результат:	2	3

Ключевые термины

- Связная область - область, соответствующая объединению отрезков.

Краткие итоги

При решении задач, в условиях которых озвучено наступление и окончание какого-то события (открытие-закрытие скобок, приход-уход сторожа, начало-конец отрезка) используется дополнительный массив "флажков". При наступлении события "флажок" принимает значение "1", при окончании - "-1".

При решении некоторых задач (например, на нахождение "связных" областей из отрезков на координатной прямой) потребуется использовать типовой алгоритм сортировки одномерного массива.

Набор для практики

Вопросы.

- Каким образом можно отметить элементы в наборе разнообразных данных, соответствующие разным событиям?
- При необходимости сортировки данных - каким образом учитывается событие, связанное с этим данным?

- Сколько связных областей дадут два отрезка, имеющие одну общую координату (начало одного совпадает с концом другого)?
- Какое количество случаев неохраняемости галереи выдаст программа, если время ухода одного из сторожей совпадает с временем прихода его сменщика?

Упражнения.

- В четырех различных кинотеатрах идут сеансы интересных фильмов. Некий зритель захотел побывать на всех четырех сеансах. Помогите установить - сможет ли зритель попасть на все 4 фильма, если известно время начала и конца каждого фильма, при этом временем "перехода" из кинотеатра в кинотеатр можно пренебречь.
- В строке символов находится программа на языке Паскаль (записанная в одну строку). Выяснить - нет ли ошибок в расстановке ключевых слов Begin и End в программе?

Решение всех рассмотренных в лекции задач зависит от способа заполнение двумерного массива. Этот способ рассматривается в задаче "Арифметический квадрат" и является базовым для решения последующих задач. **Цель лекции:** научиться применять изученный метод при решении классических задач.

Содержание

Рассмотрим задачу.

Задача 1: "Арифметический квадрат".

Сколько путей ведет из клетки (1, 1) в клетку (N, M), с условием, что двигаться можно только вниз или вправо?

Идея решения:

1. Из клетки (1,1) в любую клетку первой строки и любую клетку первого столбца ведет только один путь (по первой строке - вправо и по первому столбцу - вниз). Заполняем единицами (обозначающими единственный путь) строку и столбец (рис. 8.1).

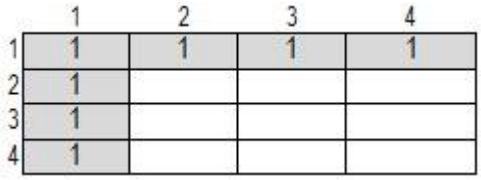

Рис. 8.1.

2. В клетку (2,2) из клетки (1,1) ведут 2 пути (рис.8.2):

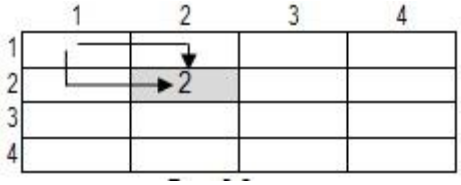

Рис. 8.2.

3. В клетку (2,3) ведут 3 пути (рис.8.3):

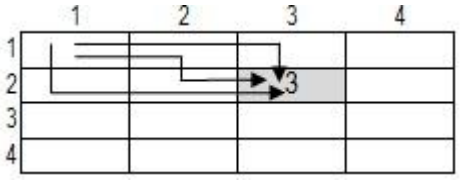

Рис. 8.3.

4. и т.д.
5. Количество путей, ведущих в клетку из клетки (1,1) зависит от значений верхнего и левого элементов массива (рис.8.4):

Итого: в клетку (4, 4) ведет 20 путей.

Решение задачи на Бейсике:

```
input "введите размерность массива", n, m
dim a (n,m)
for j=1 to m
        a(1,j)=1
next
for i=1 to n
        a(i,1)=1
next
for i=2 to n
for j=2 to m
        a(i,j)=a(i-1,j)+a(i,j-1)
next j,i
print "количество путей="; a(n,m)
```

Решение задачи на Паскале:

```
const nn=10;
        mm=10;
var     a: array [1..nn,1..mm] of integer;
        n,m,i,j: integer;
begin
        writeln ('введите размерность массива');
        readln (n,m);
        for j:=1 to m do
                a[1,j]:=1;
        for i:=1 to n do
                a[i,1]:=1;
        for i:=2 to n do
                for j:=2 to m do
                        a[i,j]:=a[i-1,j]+a[i,j-1];
        writeln (a [n,m]);
end.
```

Задача 2: Выведите на экран "**Треугольник Паскаля**" (рис.8.4):

	1	2	3	4
1		1	1	1
2	1	1+1=2	2+1=3	3+1=4
3	1	2+1=3	3+3=6	6+4=10
4	1	1+3=4	4+6=10	10+10=20

Рис. 8.4.

Идея решения:

Треугольник Паскаля расположен выше и на побочной диагонали Арифметического квадрата:

Решение задачи на Бейсике:

```
input "введите размерность"; n
dim a (n,n)
for i=1 to n
  a(1,i)=1
  a(i,1)=1
next
for i=2 to n
  for j=2 to n-i+1
    a(i,j)=a(i-1,j)+a(i,j-1)
next j,i
for i=1 to n
  for j=1 to n-i+1
    print a(i,j);
  next j
  print
next i
```

Решение задачи на Паскале:

```
const   nn=10;
var a: array [1..nn,1..nn] of integer;
  n,i,j: integer;
begin
  writeln ('введите размерность массива');
  readln (n);
  for j:=1 to n do
    begin
        a[1,j]:=1;
        a[i,1]:=1;
        end;
  for i:=2 to n do
        for j:=2 to m
          a[i,j]:=a[i-1,j]+a[i,j-1];
  for i:=1 to n do
        begin
        for j:=1 to (n-i+1) do
          write (a[i,j]);
```

```
        writeln;
        end;
end.
```

Задача 3: Раскрыть скобки в выражении: $(A+B)^n$ (n ввести с клавиатуры).

Дополнительные сведения:

"Бином Ньютона":

$$(A+B)^2 = A^2B^0 + 2A^1B^1 + A^0B^2$$
$$(A+B)^3 = A^3B^0 + 3A^2B^1 + 3A^1B^2 + A^0B^3$$
$$(A+B)^4 = A^4B^0 + 4A^3B^1 + 6A^2B^2 + 4A^1B^3 + A^0B^4$$
$$(A+B)^5 = A^5B^0 + 5A^4B^1 + 10A^3B^2 + 10A^2B^3 + 5A^1B^4 + A^0B^5$$

и т. д.

Идея решения:

- Коэффициенты при слагаемых находятся НА ОСНОВАНИИ ТРЕУГОЛЬНИКА ПАСКАЛЯ (на побочной диагонали Арифметического квадрата, размерностью (n+1)x(n+1)).
- Степень A уменьшается, степень B увеличивается от 0 до n.

Решение задачи на Бейсике:

```
input "введите степень", st
n=st+1
dim a (n,n)
for i=1 to n
  a(1,i)=1
  a(i,1)=1
next
for i=2 to n
  for j=2 to n-i+1
    a(i,j)=a(i-1,j)+a(i,j-1)
next j,i
for i=1 to n
  print a(i,n-i+1); "*a^"; n-i; "*b^"; i-1; "+";
next i
```

Решение задачи на Паскале:

```
const m=10;
var a: array [1..m,1..m] of integer;
  st,n,i,j: integer;
begin
  writeln ('введите степень ');
  readln (st);
  n:=st+1;
  for i:=1 to n do
    begin
        a[1,i]:=1;
        a[j,1]:=1;
    end;
  for i:=2 to n do
    for j:=2 to (n-i+1) do
```

```
          a[i,j]:=a[i-1,j]+a[i,j-1];
  for i:=1 to n do
       write (a[i,n-i+1], '*a^', n-i, '*b^', i-1, '+');
end.
```

Задача 4: Найти кратчайший путь в двумерном массиве из клетки 1,1 в клетку N, M. (путем называется набор индексов проходимых элементов, кратчайшим путем - набор индексов тех элементов массива, сумма значений которых минимальна). Двигаться можно только вниз либо вправо.

Идея решения:

Допустим, дан массив (рис.8.5):

	1	2	3	4
1	1	1	1	1
2	1	2	3	4
3	1	3	6	10
4	1	4	10	20

Рис. 8.5.

- Предположим, что мы будем двигаться от клетки (1,1) вправо по первой строке. Мы уже выяснили, что в каждую клетку первой строки ведет только один путь (см. "Арифметический квадрат"). Тогда пройденный к каждой клетке путь будет вычисляться как сумма значения данного элемента и элемента, стоящего левее (рис.8.6):

	1	2	3	4
1	1	4	2	1
2	5	3	1	4
3	2	1	3	5

Рис. 8.6.

- Аналогично заполняем первый столбец (рис.8.7):

	1	2	3	4
1	1	5(1+4)	7(5+2)	8(7+1)
2	5	3	1	4
3	2	1	3	5

Рис. 8.7.

- В клетку (2,2) можно попасть двумя путями - из клетки, стоящей выше или клетки, стоящей левее. Анализируем значения этих клеток (клетки, стоящей выше и клетки, стоящей левее). Из той клетки, значение которой меньше мы и придем в (2,2). Значит, значение клетки (2,2) увеличивается на 5 (рис.8.08):

	1	2	3	4
1	1	5	7	8
2	6(1+5)	3	1	4
3	8(6+2)	1	3	5

Рис. 8.08.

- Аналогично заполняем остальные элементы массива - анализируем содержимое элемента, стоящего сверху и элемента слева. То значение, которое меньше и прибавляем к содержимому текущего элемента.
- Вычисление координат: Движемся в обратном направлении: из клетки (n,m) к клетке (1,1). Анализируем содержимое элемента, стоящего левее и выше текущего. Координаты того элемента, значение которого меньше и выводим на экран (рис.8.09):

	1	2	3	4
1	**1**	**5**	**7**	8
2	6	8	**8**	12
3	8	9	**11**	**16**

Рис. 8.09.

Итого: Кратчайший путь - (3,4)-(3,3)-(2,3)-(1,3)-(1,2)-(1,1)

Решение задачи на Бейсике:

```
input "введите размерность массива", n, m
dim a(n,m)
for i=1 to n
  for j=1 to m
    input "введите элемент"; a(i,j)
  next j
next i
rem================================================
for j=2 to m
  a(1,j)= a(1,j)+ a(1,j-1)
next
for i=2 to n
  a(i,1)= a(i,1)+ a(i-1,1)
next
for i=2 to n
  for j=2 to m
        if a(i-1,j)<a(i,j-1) then a(i,j)=a(i,j)+a(i-1,j) else
a(i,j)=a(i,j)+a(i,j-1)
  next j
next i
rem= вывод координат в обратном порядке=====================
i=n
j=m
print  n,m
for x=1 to n+m-3
  if a(i-1,j)<a(i,j-1) then i=i-1 else j=j-1
  if i<>0 and j<>0 then print i;"-:;j
next
print 1,1
```

Решение задачи на Паскале:

```
const nn=10;
mm=10;
var a: array [1..nn,1..mm] of integer;
  i,j,n,m,x: integer;
begin
  writeln ('введите размерность массива');
  readln (n, m);
  for i:=1 to n do
    for j:=1 to m do
          readln (a[i,j]);
  for j:=2 to m do
     a [1,j]:= a [1,j]+ a [1,j-1];
  for i:=2 to n do
     a [i,1]:= a [i,1]+ a [i-1,1];
  for i:=2 to n do
         for j:=2 to m do
          if a [i-1,j]<a [i,j-1] then a [i,j]:=a [i,j]+a [i-1,j]
          else a [i,j]:=a [i,j]+a [i,j-1];
  i:=n;
  j:=m;
  {======вывод координат в обратном порядке======}
  writeln (n,m);
  for x:=1 to n+m-3 do
    begin
         if a [i-1,j]<a [i,j-1] then i:=i-1
         else j:=j-1;
         if (i<>0) and (j<>0) then writeln (i,j);
         end;
  writeln (1,1);
end.
```

Ключевые термины

- Арифметический квадрат - двумерный массив, заполненный так: единичные элементы располагаются в первой строке и первом столбце. Остальные элементы заполняются суммой выше и левее стоящего элемента.
- Треугольник Паскаля - образован элементами, расположенными на и выше побочной диагонали арифметического квадрата.
- Бином Ньютона - формула для разложения на отдельные слагаемые целой неотрицательной степени суммы двух переменных (формула разложения произвольной натуральной степени двучлена $(a+b)^n$ в многочлен).
- Путь в двумерном массиве - набор индексов проходимых элементов, кратчайшим путь - набор индексов тех элементов массива, сумма значений которых минимальна.

Краткие итоги

Способ заполнения Арифметического квадрата: единицами заполняются элементы первой строки и первого столбца. Остальные элементы заполняются суммой выше и левее стоящего элемента.

Элементы, расположенные на и выше побочной диагонали Арифметического квадрата образуют треугольник Паскаля.

На побочной диагонали Арифметического квадрата располагаются элементы, которые являются коэффициентами для слагаемых при разложении целой неотрицательной степени суммы двух переменных в Биноме Ньютона.

Нахождение кратчайшего пути в двумерном массиве также базируется на алгоритме создания Арифметического квадрата.

Набор для практики

Вопросы.

- Какое значение находится в элементе последней строки и последнего столбца Арифметического квадрата (какую информацию несет это число)?
- Пусть заполнение первой строки и первого столбца двумерного массива единицами идет в одном теле цикла. Каковы будут индексы элементов первой строки и индексы элементов первого столбца?
- Массив какой размерности необходимо заполнить, что бы получить коэффициенты для слагаемых при разложении суммы двух переменных в четвертой степени?

Упражнения.

- Вывести на экран Арифметический квадрат размерностью NxN (N ввести с клавиатуры).
- Вывести на экран коэффициенты для слагаемых при разложении суммы двух переменных в десятой степени.

Направление (порядок) обхода элементов двумерного массива может пригодиться в решениях некоторых задач повышенной сложности, например, в решениях таких задач, как "Магический квадрат" и "Скатерть Улама". В решении задачи "Скатерть Улама" используется пройденный на предыдущих лекциях алгоритм "Решето Эратосфена". При решении всех задач данной лекции используются ранее рассмотренные типовые алгоритмы обработки двумерных массивов. **Цель лекции:** научиться применять изученный метод при решении классических задач.

Содержание

Рассмотрим задачу.

Задача 1: Заполнить двумерный массив размерностью NxN, как показано на рис.9.1 (ряд натуральных чисел указывает на направление обхода элементов):

	1	2	3	4
1	1	2	4	7
2	3	5	8	
3	6	9		
4	10			

Рис. 9.1.

Идея решения: Мысленно "разберем" двумерный массив на четыре "вложенных" в исходный (рис.9.2). В каждом из них нужно заполнить ПОБОЧНУЮ ДИАГОНАЛЬ (опираясь на типовой алгоритм обработки квадратного массива относительно диагоналей):

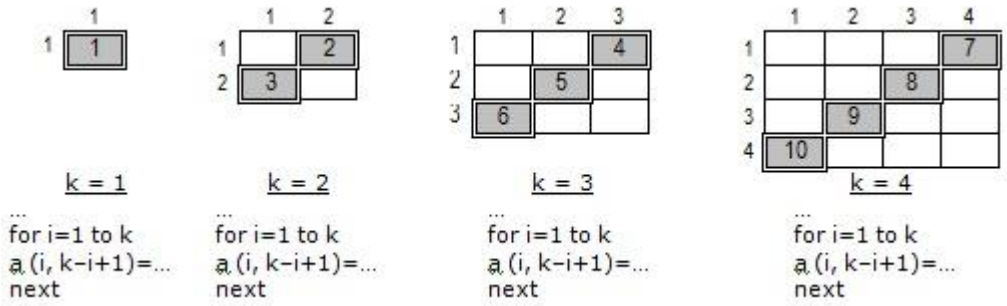

Рис. 9.2.

Итого, k изменяется от 1 до n:

```
for k=1 to n
```

```
 for i=1 to k
  a (i,k-i+1)= …
 next  i
next k
```

Полное решение задачи на Бейсике:

```
input "n="; n
dim a(n,n)
x=1
for k=1 to n
 for i=1 to k
  a (i,k-i+1)= x
  x=x+1
 next i
next k
rem=====================
for i=1 to n
 for j=1 to n
  print a (i,j);
 next j
 print
next i
```

Полное решение задачи на Паскале:

```
const m=10;
var a: array [1..m, 1..m] of byte;
 x, n, k, i: integer;
begin
 writeln ('n='); readln (n); x:=1;
 for k:=1 to n do
  for i:=1 to k do
   begin
   a [i,k-i+1]:=x;
   x:=x+1;
   end;
 for i:=1 to n do
  begin
  for j:=1 to n do write (a [i,j]);
  writeln;
  end;
end.
```

Задача 2: Заполнить массив, как показано на табл.9.1:

Таблица 9.1.

1	1	1	1	1	1
1	2	2	2	2	2
1	2	3	3	2	1
1	2	3	3	2	1
1	2	2	2	2	1
1	1	1	1	1	1

Идея решения: Разложим исходную матрицу на "вложенные" (рис.9.3):

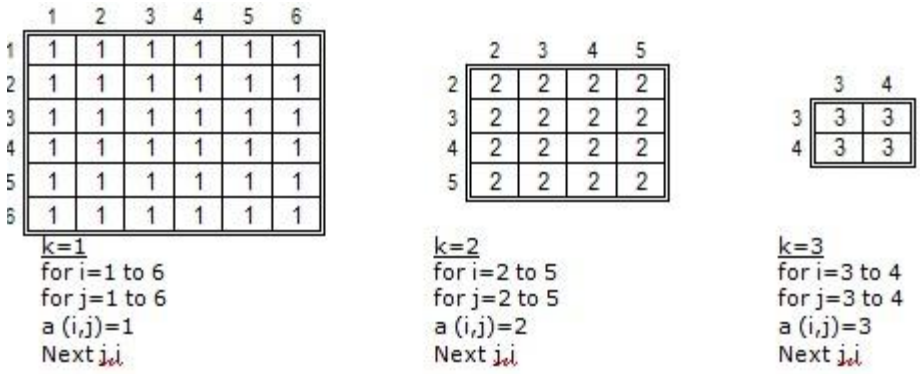

Рис. 9.3.

Решение задачи на Бейсике:

```
Input "n="; n
Dim a(n,n)
for k=1 to n\2+1
```

```
 for i=k to n-k+1
  for j=k to n-k+1
   a (i,j)= k
  next j
 next i
next k
rem=====вывод======
for i=1 to n
 for j=1 to n
  print a (i,j);
 next j
 print
next i
```

Решение задачи на Паскале:

```
const m=10;
var a: array [1..m, 1..m] of byte;
 x, n, k, I, j: integer;
begin
 writeln ('n=');
 readln (n);
 for k:=1 to (n div 2 +1) do
  for i:=k to n-k+1 do
   for j:=k to n-k+1 do
        a [i,j]:= k;
 {=======вывод=========}
 for i:=1 to n do
  begin
  for j:=1 to n do
   write (a [i,j]);
  writeln;
  end;
end.
```

Задача 3: Заполнить массив, как показано на табл.9.2:

Таблица 9.2.

1	2	3	4	5	6
20	21	22	23	24	7
19	32	33	34	25	8
18	31	36	35	26	9
17	30	29	28	27	10
16	15	14	13	12	11

Дополнительные сведения: На таком заполнении массива базируется задача "скатерть Улама". В данной задаче квадратный массив заполняется по спирали (заполнение происходит не "вовнутрь", как в примере, а "изнутри" массива). Затем "вычеркиваются" все составные числа (см. задачу "Решето Эратосфена"). Оставшиеся на своих местах простые числа образуют причудливый узор, названный в честь автора "Скатертью Улама".

Идея решения: Разложим исходную матрицу на "вложенные". Каждую из "вложенных" матриц необходимо заполнить по периметру (рис.9.4).

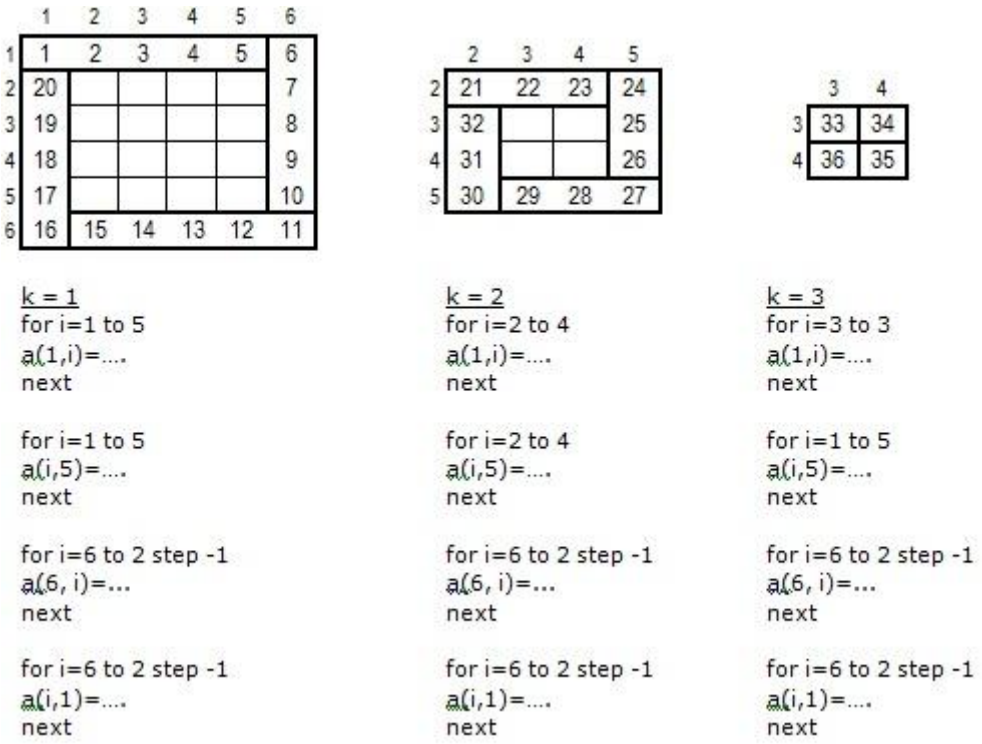

Рис. 9.4.

Решение на Бейсике:

```
input "n="; n
dim a (n,n)
x=1
for k=1 to n\2
 for i=k to n-k
  a(k,i)=x
  x=x+1
 next
 rem=============
 for i=k to n-k
  a(i,n-k+1)=x
  x=x+1
 next
 rem=============
for i=k to n-k
  a(n-k+1,n-i+1)=x
  x=x+1
 next
 rem=============
for i=k to n-k
  a(n-i+1,k)=x
  x=x+1
 next i
next k
rem=================
for i=1 to n
 for j=1 to n
  print a(i,j);
 next j
 print
next i
```

Решение на Паскале:

```
const m=10;
var a: array [1..m, 1..m] of byte;
 x, n, k, i, j: integer;
begin
 writeln ('n='); readln (n);
 x:=1;
 for k:=1 to n div 2 do
  begin
  for i:=k to n-k do
    begin
        a [k,i]:=x;
        x:=x+1;
        end;
  for i:=k to n-k do
        begin
        a [i,n-k+1]:=x;
        x:=x+1;
        end
  for i:=k to n-k do
        begin
        a [n-k+1,n-i+1]:=x;
        x:=x+1;
        end ;
  for i:=k to n-k do
        begin
        a [n-i+1,k]:=x;
        x:=x+1;
        end;
  end;
 {=======вывод========}
 for i:=1 to n do
  begin
  for j:=1 to n do write (a[i,j]);
  writeln;
  end;
end.
```

Задача 4: Заполните матрицу, как показано на рис.9.5:

	1	2	3	4	5	6	7	8	9
1					1				
2				2		6			
3			3		7		11		
4		4		8		12		16	
5	5		9		13		17		21
6		10		14		18		22	
7			15		19		23		
8				20		24			
9					25				

Рис. 9.5.

Дополнительные сведения: Есть в информатике классическая задача "**Магический квадрат**" (в "Магическом квадрате" сумма элементов всех строк, всех столбцов и всех диагоналей равна). Решать ее можно различными способами, один из которых называется методом "Террас". Для создания Магического квадрата размерностью n x n (n - нечетное число) необходимо заполнить двумерный массив размерностью (2n-1)x(2n-1) так, как в на рис.9.6:

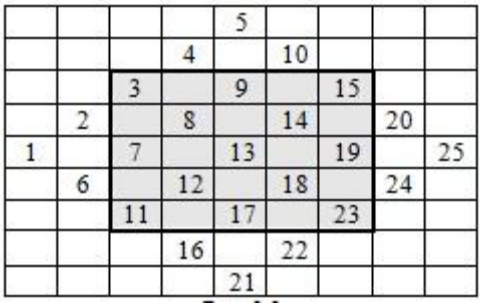

Рис. 9.6.

Затем "треугольники", выступающие за пределы жирной рамки перенести внутрь таким образом (табл.9.3):

Таблица 9.3.

3	**16**	9	**22**	15
20	8	**21**	14	**2**
7	**25**	13	**1**	19
24	12	**5**	18	**6**
11	**4**	17	**10**	23

В полученном двумерном массиве сумма элементов всех строк и всех столбцов равна.

Идея решения: Разбиваем исходную матрицу на "вложенные". В каждой матрице заполняем побочную диагональ:

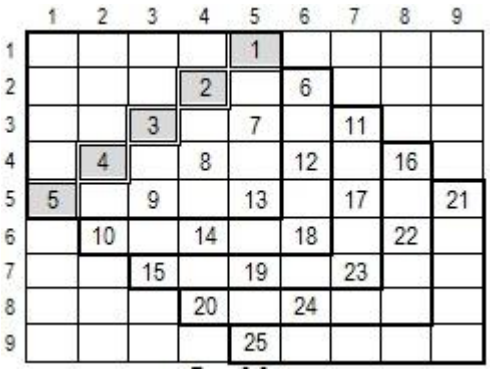

Рис. 9.7.

Решение очевидно.

Ключевые термины

- Магический квадрат - двумерный массив, сумма элементов каждой строки, каждого столбца и каждой диагонали которого одинакова.
- Метод "Террас" - способ создания магического квадрата
- Решето Эратосфена - алгоритм нахождения простых чисел.
- Скатерть Улама - узор из простых чисел, расположенных по спирали.

Краткие итоги

При заполнении некоторых двумерных массивов проглядывается некоторая закономерность (способ заполнения повторяется). Мысленно "разберем" двумерный массив на "вложенные" - как бы независимые друг от друга массивы, для заполнения которых используется один и тот же способ. Разрабатываем для каждого из них алгоритм заполнения, затем находим зависимость, объединяющую все алгоритмы в один.

Данный способ поможет решить такие задачи, как создание Магического квадрата методом "Террас", создание "Скатерти Улама" при помощи "Решета Эратосфена".

Набор для практики

Вопросы.

- Каким образом можно заполнить двумерный массив числами натурального ряда чисел?
- Каков порядок обхода элементов двумерного массива, если счетчик внешнего цикла используется в качестве первого индекса элемента массива, счетчик внутреннего цикла - в качестве второго индекса?
- Каков порядок обхода элементов двумерного массива, если счетчик внешнего цикла используется в качестве второго индекса элемента массива, счетчик внутреннего цикла - в качестве первого индекса?

Упражнения.

- Заполните массив по образцу (табл.9.4):

Таблица 9.4.

1	2	3	4	5	6	7
14	13	12	11	10	9	8
15	16	17	18	19	20	21
28	27	26	25	24	23	22
и т.д.	...	...	...	...	...	...
...	...					

- Заполните массив по образцу (табл.9.5):

Таблица 9.5.

1	2	6	7	15	16	28
3	5	8	14	17	27	
4	9	13	18	26		
10	12	19	25			
11	20	24				
21	23					
22						

- Заполните массив по образцу (табл. 9.6):

Таблица 9.6.

22	16	11	7	4	2	1
	23	17	12	8	5	3
		24	18	13	9	6
			25	19	14	10
				26	20	15
					27	21
						28

На вычислении площади треугольника базируются многие геометрические задачи. В лекции рассматриваются примеры геометрических задач, в основе решения которых лежит вычисление площади треугольника. **Цель лекции:** научиться применять изученные методы при решении геометрических задач.

Содержание

Рассмотрим решение базовой задачи.

Задача : Найти площадь треугольника, заданного координатами своих вершин (рис.10.1).

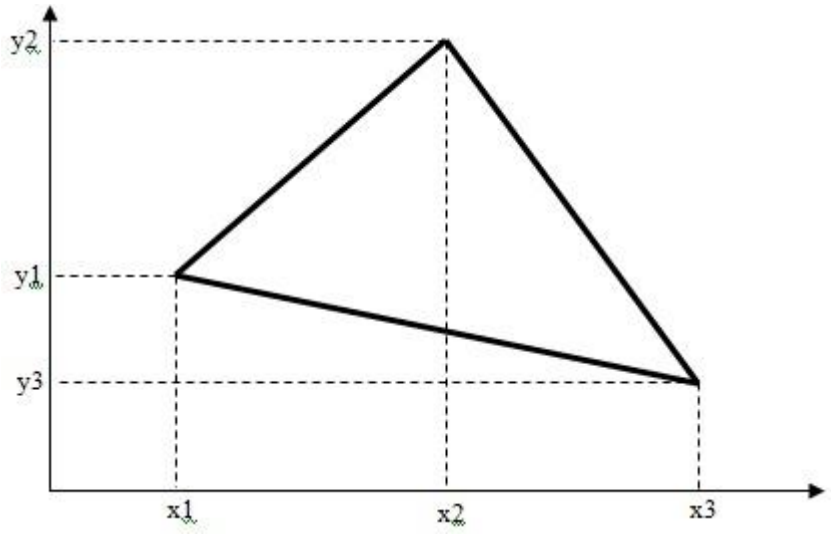

Рис. 10.1.

Справочная информация: Для нахождения площади треугольника воспользуемся **формулой Герона**:

S=SQR (p(p-a)(p-b)(p-c))

где a, b, c - длины сторон, а p - полупериметр.

Программа на Бейсике:

```
print "введите пары координат трех точек"
input x1, y1
input x2, y2
input x3, y3
a= sqr ((x1-x2)^2+(y1-y2)^2)
```

```
b= sqr ((x2-x3)^2+(y2-y3)^2)
c= sqr ((x3-x1)^2+(y3-y1)^2)
per= a+b+c
print  "периметр треугольника="; per
p=per/2
s= sqr (p*(p-a)*(p-b)*(p-c))
print "площадь="; s
```

Программа на Паскале:

```
var x1,x2,x3,y1,y2,y3: integer;
  a,b,c,p: real;
begin
  writeln ('введите пары координат трех точек');
  readln (x1, y1);
  readln (x2, y2);
  readln (x3, y3);
  a:= sqrt (sqr(x1-x2)+sqr(y1-y2));
  b:= sqrt (sqr(x2-x3)+sqr(y2-y3));
  c:= sqrt (sqr(x3-x1)+sqr(y3-y1));
  p:=(a+b+c) / 2;
  writeln ('площадь треугольника:=', sqrt (p*(p-a)*(p-b)*(p-c)));
end.
```

Тест:

Дано:	1,2 1,5 5,2
Результат:	6

Задача: Определить площадь выпуклой фигуры, заданной координатами своих вершин (координаты вводятся в соответствии с последовательным обходом вершин - см. рис.10.2).

Идея решения: задачу можно решить двумя способами.

1 способ: площадь выпуклой фигуры равна сумме площадей треугольников, имеющих общую вершину:

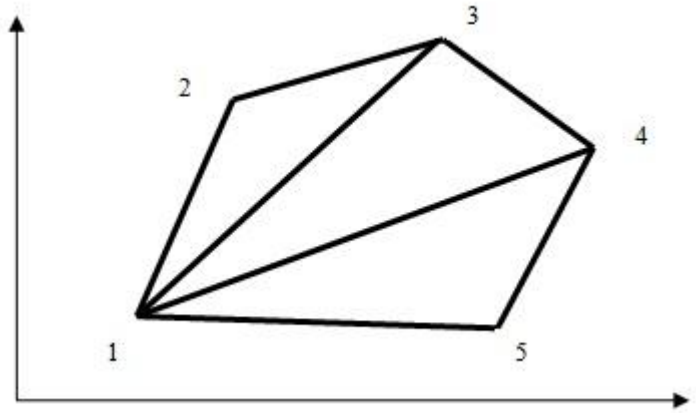

Рис. 10.2.

Программа на Бейсике:

```
input "введите количество вершин многоугольника"; n
dim x(n), y(n)
for i=1 to n
  input "введите пару координат"; x(i), y(i)
next
for i=2 to n-1
  a= sqr ((x(1)-x(i))^2+(y(1)-y(i))^2)
  b= sqr ((x(i)-x(i+1))^2+(y(i)-y(i+1))^2)
  c= sqr ((x(i+1)-x(1))^2+(y(i+1)-y(1))^2)
  p= (a+b+c)/2
  s=s+sqr (p*(p-a)*(p-b)*(p-c))
next
print "площадь фигуры="; s
```

Программа на Паскале:

```
const nn=10 ;
var x,y: array [1..nn] of integer;
  i, n: integer;
  a, b, c, s, p: real;
begin
  writeln ('введите количество вершин многоугольника');
  readln (n);
  for i:=1 to n do
    begin
        writeln (''введите пару координат');
        readln (x[i], y[i]);
        end;
  for i:=2 to n-1 do
    begin
        a:= sqrt (sqr(x[1]-x[i])+sqr(y[1]-y[i]));
        b:= sqrt (sqr(x[i]-x[i+1])+sqr(y[i]-y[i+1]));
        c:= sqrt (sqr(x[i+1]-x[1])+sqr(y[i+1]-y[1]));
        p:= (a+b+c)/2
        s:=s+sqrt (p*(p-a)*(p-b)*(p-c))
        end;
  writeln ('площадь фигуры=', s);
end.
```

Тест:

Дано:	N=5 3,2 2,5 4,7 8,5 7,1
Результат:	23.5

2 способ: Второй способ вычисления площади выпуклой фигуры иллюстрирует схема (рис. 10.3):

Рис. 10.3.

Решите задачу этим методом самостоятельно.

Задача 3: Определить - находится точка внутри или вне выпуклого многоугольника, заданного координатами своих вершин.

Идею решения задачи иллюстрирует схема (рис.10.4):

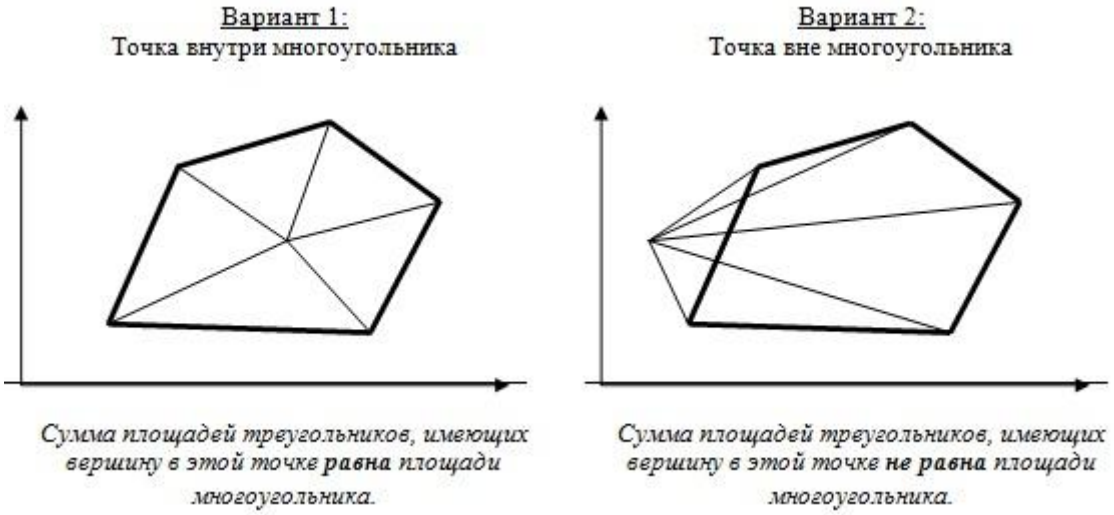

Рис. 10.4.

Программа на Бейсике:

```
input "введите координаты точки"; x, y
input "введите количество вершин многоугольника"; n
dim x(n+1), y(n+1)
for i=1 to n
  input "введите координаты вершин"; x(i), y(i)
next
x(n+1)=x(1)
y(n+1)=y(1)
rem=====================================
for i=2 to n-1
  a= sqr ((x(1)-x(i))^2+(y(1)-y(i))^2)
  b= sqr ((x(i)-x(i+1))^2+(y(i)-y(i+1))^2)
  c= sqr ((x(i+1)-x(1))^2+(y(i+1)-y(1))^2)
  p= (a+b+c)/2
  s1=s1+sqr (p*(p-a)*(p-b)*(p-c))
next
rem=====================================
for i=1 to n
  a= sqr ((x-x(i))^2+(y-y(i))^2)
  b= sqr ((x(i)-x(i+1))^2+(y(i)-y(i+1))^2)
  c= sqr ((x(i+1)-x)^2+(y(i+1)-y)^2)
  p= (a+b+c)/2
  s2=s2+sqr (p*(p-a)*(p-b)*(p-c))
next
if s1=s2 then print "точка внутри" else print "точка вне фигуры"
```

Программа на Паскале:

```
const nn=100 ;
var x,y: array [1..nn] of integer;
  i,xx,yy,n: integer;
  a,b,c, p, s1,s2: real;
begin
  writeln ('введите координаты точки');
  readln (xx,yy);
  writeln ('введите количество вершин многоугольника');
  readln (n);
  for i:=1 to n do
    begin
        writeln ('введите координаты вершин');
        readln (x[i], y[i]);
        end;
  x[n+1]:=x[1];
  y[n+1]:=y[1];
  for i:=2 to n-1 do
    begin
        a:= sqrt (sqr(x[1]-x[i])+sqr(y[1]-y[i]));
        b:= sqrt (sqr(x[i]-x[i+1])+sqr(y[i]-y[i+1]));
        c:= sqrt (sqr(x[i+1]-x[1])+sqr(y[i+1]-y[1]));
        p:= (a+b+c) / 2;
        s1:=s1+sqrt (p*(p-a)*(p-b)*(p-c));
        end;
  for i:=1 to n do
        begin
        a:= sqrt (sqr(xx-x[i])+sqr(yy-y[i]));
        b:= sqrt (sqr(x[i]-x[i+1])+sqr(y[i]-y[i+1]));
        c:= sqrt (sqr(x[i+1]-xx)+sqr(y[i+1]-yy));
        p:= (a+b+c) / 2;
        s2:=s2+sqrt (p*(p-a)*(p-b)*(p-c));
        end;
  if round(s1)=round(s2) then writeln ('точка внутри')
  else writeln ('точка вне фигуры');
end.
```

Тест:

Дано:	5,4 5 3,2 2,5 4,7 8,5 7,1	10,10 5 3,2 2,5 4,7 8,5 7,1
Результат:	внутри	вне

Задача 4: Определить, пересекаются ли два отрезка, заданные на плоскости координатами своих концов (примеры взаимного расположения отрезков на рис.10.5).

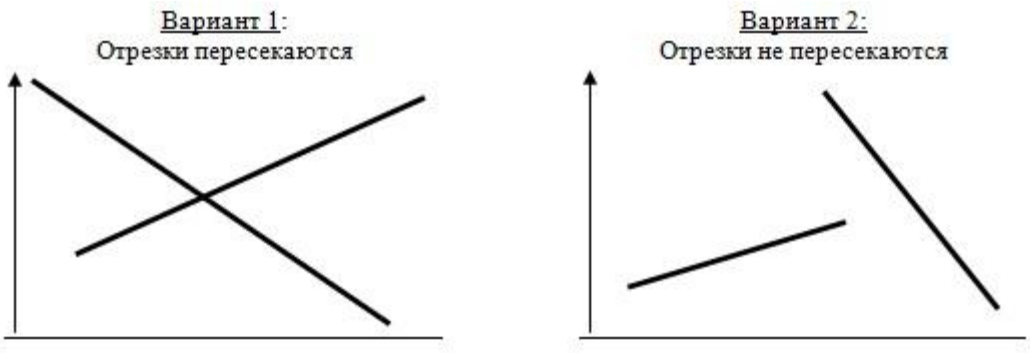

Рис. 10.5.

Идею решения задачи иллюстрирует схема (рис.10.6):

1. Отрезки пересекаются если:

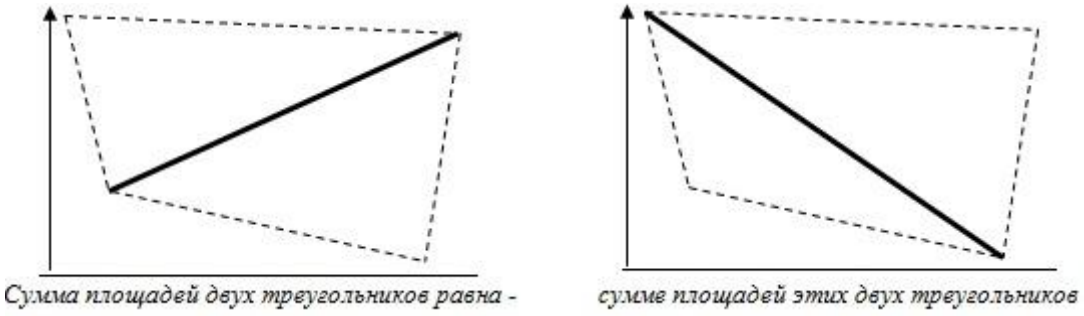

Рис. 10.6.

2. Отрезки не пересекаются (рис.10.7) если …

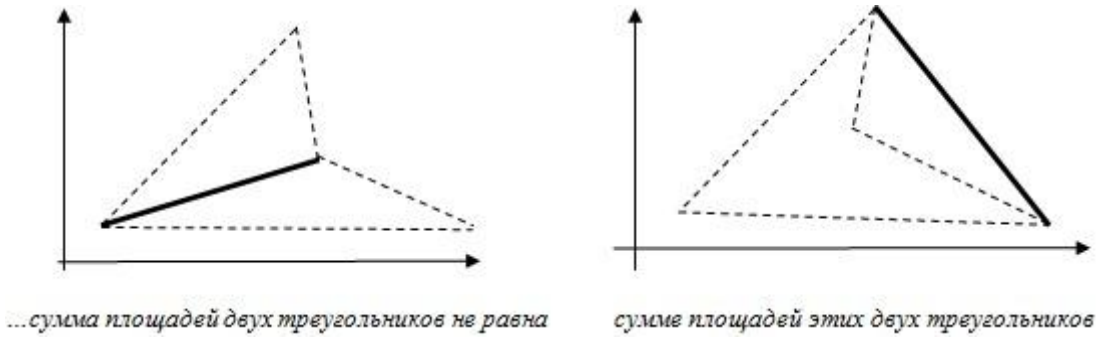

Рис. 10.7.

Следует отметить, что данный метод определения пересечения двух отрезков имеет исключение, в случаях:

- если отрезки равной длины и параллельны;
- если отрезки находятся на одной прямой, но не имеют общих точек

Программа на Бейсике:

```
input "введите две координаты концов 1 отрезка"; x1,y1
input "введите две координаты концов 1 отрезка"; x2,y2
input "введите две координаты концов 2 отрезка"; xx1,yy1
input "введите две координаты концов 2 отрезка"; xx2,yy2
rem=площадь первой фигуры======================
a=sqr ((xx1-xx2)^2+(yy1-yy2)^2)
b=sqr ((x1-xx1)^2+(y1-yy1)^2)
c=sqr ((x1-xx2)^2+(y1-yy2)^2)
p= (a+b+c)/2
s1=s1+sqr (p*(p-a)*(p-b)*(p-c))
rem
b=sqr ((x2-xx1)^2+(y2-yy1)^2)
c=sqr ((x2-xx2)^2+(y2-yy2)^2)
p= (a+b+c)/2
s1=s1+sqr (p*(p-a)*(p-b)*(p-c))
rem=площадь второй фигуры=========================
a=sqr ((x1-x2)^2+(y1-y2)^2)
b=sqr ((x2-xx1)^2+(y2-yy1)^2)
c=sqr ((x1-xx1)^2+(y1-yy1)^2)
p= (a+b+c)/2
s2=s2+sqr (p*(p-a)*(p-b)*(p-c))
rem
b=sqr ((x1-xx2)^2+(y1-yy2)^2)
c=sqr ((x2-xx2)^2+(y2-yy2)^2)
p= (a+b+c)/2
s2=s2+sqr (p*(p-a)*(p-b)*(p-c))
rem=========================================
print s1,s2
if int(s1)=int(s2) then print  "пересекаются" else print "не пересекаются"
```

Программа на Паскале:

```
var x1,x2,y1,y2,xx1,xx2,yy1,yy2: integer;
  p,a,b,c,s1,s2:real;
begin
  writeln ('введите две координаты концов 1 отрезка');
  readln (x1,y1);
  writeln ('введите две координаты концов 1 отрезка');
  readln (x2,y2);
  writeln ('введите две координаты концов 2 отрезка');
  readln (xx1,yy1);
  writeln ('введите две координаты концов 2 отрезка');
  readln (xx2,yy2);
  {=====площадь первой фигуры============}
  a:=sqrt (sqr(xx1-xx2)+sqr(yy1-yy2));
  b:=sqrt (sqr(x1-xx1)+sqr(y1-yy1));
  c:=sqrt (sqr(x1-xx2)+sqr(y1-yy2));
  p:= (a+b+c)/2;
  s1:=s1+sqrt (p*(p-a)*(p-b)*(p-c));
  {================================}
  b:=sqrt (sqr(x2-xx1)+sqr(y2-yy1));
  c:=sqrt (sqr(x2-xx2)+sqr(y2-yy2));
  p:= (a+b+c)/2;
  s1:=s1+sqrt (p*(p-a)*(p-b)*(p-c));
  {=====площадь второй фигуры============}
  a:=sqrt (sqr (x1-x2)+ sqr (y1-y2));
  b:=sqrt (sqr (x2-xx1)+ sqr (y2-yy1));
  c:=sqrt (sqr (x1-xx1)+ sqr (y1-yy1));
  p:= (a+b+c)/2;
  s2:=s2+sqrt (p*(p-a)*(p-b)*(p-c));
  {================================}
  b:=sqrt (sqr (x1-xx2)+ sqr (y1-yy2));
  c:=sqrt (sqr (x2-xx2)+ sqr (y2-yy2));
```

```
  p:= (a+b+c)/2;
  s2:=s2+sqrt (p*(p-a)*(p-b)*(p-c));
  {================================}
  writeln (s1,s2);
  if round(s1)=round(s2) then writeln ('пересекаются')
  else writeln ('не пересекаются');
end.
```

Тест:

Дано:	2,4 8,7 3,8 8,1	3,6 8,1 6,5 11,7
Результат:	Пересекаются	Не пересекаются

Ключевые термины

- Формула Герона - формула для вычисления площади треугольника по длинам его сторон.

Краткие итоги

Для определения площади треугольника необходимо воспользоваться формулой Герона.

Для нахождения площади многоугольника потребуется найти сумму площадей суммы треугольников, составляющих этот многоугольник.

Решение задач на определение вхождения точки в область многоугольника, пересечения двух отрезков необходимо также найти площади треугольников, составляющих фигуры.

Набор для практики

Вопросы.

- Какой формулой пользуются для вычисления площади треугольника?
- Каким способом вычисляют площадь многоугольника?
- Опишите метод для определения вхождения точки внутрь многоугольника.
- Подумайте, как можно определить "вхождение" одной фигуры в другую?
- Опишите метод для определения, пересекаются ли два отрезка.
- Подумайте - есть ли исключения у метода определения пересечения двух отрезков?

Упражнения.

- Треугольник на плоскости задан координатами своих вершин. Определить - прямоугольный ли этот треугольник?
- Найти площадь 4-хугольника (не обязательно прямоугольника), заданного координатами своих вершин (на рис.10.8 изображен частный случай). Найти другой метод (кроме метода разбиения многоугольника на треугольники) нахождения площади этого прямоугольника.

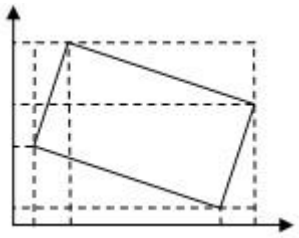

Рис. 10.8.

- С клавиатуры вводятся пары координат n точек на плоскости. Найти длину ломанной, соединяющей эти точки. Начало ломанной - ближайшая к центру координатной плоскости точка, ломанная соединяет все точки кратчайшим способом.
- Многоугольник задан координатами своих последовательных вершин. Внутренний угол в одной из вершин многоугольника - тупой. Определите эту вершину.

Комбинаторные задачи бесспорные лидеры на олимпиадах. Поэтому этой теме следует уделить особое внимание. Начинать изучение данной темы нужно с азов. В лекции рассмотрен первый класс задач, в которых ОПРЕДЕЛЕНО N и K (N - количество элементов в исходном множестве, K - количество предметов, выбираемых из исходного множества). **Цель лекции:** сформировать понятие о комбинаторных группах и способах их получения.

Содержание

Выборку по K элементов из множества A можно производить, руководствуясь правилами: считать разными выборки, в которых один и тот же элемент занимает разные позиции (либо не считать), допускать повторение одного и того же элемента в выбираемой группе (либо не допускать) и др. В зависимости от наложения определенных ограничений (правил) при выборе элементов разделяют три типа формирования комбинаторных групп. Рассмотрим их на примере:

Пусть N=4, K=2. Элементы исходного множества будем хранить в массиве A (рис.11.1):

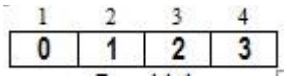

Рис. 11.1.

Тогда группы по 2 элемента, выбираемые из множества A можно сформировать так, как показано в таблице 11.1:

Таблица 11.1.

Размещения		Сочетания	
С повторениями	**Без повторений**	**С повторениями**	**Без повторений**
00	01	00	01
01	02	01	02
02	03	02	03
03	10	03	12
10	12	11	13
11	13	12	23
12	20	13	

13	21	22	
20	23	23	
21	30	33	
22	31		
23	32		
30			
31			
32			
33			
Группы {01} и {10} считаются различными	Исключаются группы, в кот. один и тот же элемент стоит в разных позициях	Группы {01} и {10} считаются одинаковыми	Исключаются группы, в кот. один и тот же элемент стоит в разных позициях

Существует еще и третий способ формирования комбинаторных групп: "Перестановки". В перестановках участвуют все элементы исходного множества (K=N). Перестановки с повторениями возможны, когда в исходном множестве есть повторяющиеся элементы.

Типовые алгоритмы формирования групп:

Размещения с повторениями:

Программная реализация на Бейсике	Программная реализация на Паскале

`for i=1 to n`
`for j=1 to n`
`print A(i), A(j);`
`next j, i`
	`for i:=1 to n do`
	`for j:=1 to n do`
	`writeln (A[i], A[j]);`

Размещения без повторений:

Программная реализация на Бейсике	Программная реализация на Паскале

`for i=1 to n`
`for j=1 to n`
`if i<>j then print A(i), A(j);`
`next j,i`
	`for i:=1 to n do`
	`for j:=1 to n do`
	`if i<>j then writeln (A[i], A[j]);`

Сочетания с повторениями:

Программная реализация на Бейсике	Программная реализация на Паскале

`...`
`for i=1 to n`
`for j=i to n`
`print A(i,j);`
`next j, i`
	`...`
	`for i:=1 to n do`
	`for j:=i to n do`
	`writeln (A[i], A[j]);`

Сочетания без повторений:

Программная реализация на Бейсике	Программная реализация на Паскале
...	...
for i=1 to n-1	for i:=1 to n-1 do
for j=i+1 to n	for j:=i+1 to n do
print A(i), A(j);	writeln (A[i], A[j]);
next j, i	

Задача "Кодовый замок сейфа"

Из 10 букв нужно набрать 3. Повторение букв допустимо. Подсчитать количество возможных комбинаций кодов.

Идея решения: Необходимо применить типовой алгоритм формирования групп РАЗМЕЩЕНИЯ С ПОВТОРЕНИЯМИ.

Программа на Бейсике

```
dim a$(10)
for i=1 to 10
  input "введите букву"; a$(i)
next
for x1=1 to 10
  for x2=1 to 10
    for x3=1 to 10
          print  a$(x1); a$(x2); a$(x3);
          k=k+1
next x3,x2,x1
print "k="; k
```

Программа на Паскале:

```
var a: array [1.10] of char;
  x1, x2, x3, k, i: integer;
begin
  for i:=1 to 10 do
    readln (a[i]);
  for x1:=1 to 10 do
    for x2:= 1 to 10 do
          for x3:=1 to 10 do
            begin
                writeln  (a[x1], a[x2], a[x3]);
                k:=k+1;
                end;
  writeln ('k:=', k);
end.
```

Тест:

Результат: 1000

Задача

В ассортименте кондитерской сегодня 10 видов пирожных. Каких три пирожных продавец может предложить покупателю?

Идея решения: Необходимо применить типовой алгоритм формирования групп СОЧЕТАНИЯ С ПОВТОРЕНИЯМИ.

Программа на Бейсике

```
dim a$ (10)
for i=1 to 10
  input "введите название пирожного"; a$ (i)
next
for x1=1 to 10
  for x2=1 to 10
    for x3=1 to 10
         print  a$(x1); a$(x2); a$(x3)
next x3,x2,x1
```

Программа на Паскале:

```
var a: array [1.10] of string;
  x1, x2, x3, i: integer;
begin
  for i:=1 to 10 do
    begin
        writeln ('введите название пирожного');
        readln (a[i]);
        end;
  for x1:=1 to 10 do
        for x2:= 1 to 10 do
          for x3:=1 to 10 do
                writeln  (a[x1], a[x2], a[x3]);
end.
```

Задачи из "Теории чисел"

Найти все 3-хзначные числа, сумма цифр которых равна К (введенному с клавиатуры)

Идея решения: Необходимо применить типовой алгоритм формирования групп РАЗМЕЩЕНИЯ С ПОВТОРЕНИЯМИ.

Программа на Бейсике

```
Input "k="; k
for x1=1 to 9
  for x2= 0 to 9
    for x3=0 to 9
          if x1+x2+x3=k then print x1; x2; x3
next x3,x2,x1
```

Программа на Паскале:

```
var x1, x2, x3, k: integer;
begin
  writeln ('k=');
  readln (k);
   for x1:=1 to 9 do
        for x2:=0 to 9 do
         for x3:=0 to 9 do
          if x1+x2+x3=k then
  writeln (x1, x2, x3);
end.
```

Тест:

Дано:	3
Результат:	102
	111
	120
	201
	210
	300

Подсчитать количество 'счастливых' троллейбусных билетов ("счастливые" номера билетов - шестизначные числа, в которых сумма первых трех цифр равна сумме вторых трех цифр).

Идея решения: Необходимо применить типовой алгоритм формирования групп РАЗМЕЩЕНИЯ С ПОВТОРЕНИЯМИ.

Программа на Бейсике:

```
for x1=1 to 9
 for x2=0 to 9
  for x3=0 to 9
   for x4=0 to 9
        for x5=0 to 9
         for x6=0 to 9
          if x1+x2+x3=x4+x5+x6 then k=k+1
next x6, x5, x4, x3, x2, x1
print "k="; k
```

Программа на Паскале:

```
var x1, x2, x3, k: integer;
begin
  k:=0;
  for x1:=1 to 9 do
   for x2:=0 to 9 do
        for x3:=0 to 9 do
         for x4:=0 to 9 do
          for x5:=0 to 9 do
           for x6:=0 to 9 do
                if x1+x2+x3=x4+x5+x6 then k:=k+1;
        writeln ('k=', k);
end.
```

Тест:

Результат:	46242

Геометрические задачи

На плоскости N точек заданы своими координатами. Найти "центральную" точку (точку, сумма расстояний от которой до остальных точек максимальна).

Идея решения: Необходимо применить типовой алгоритм формирования групп РАЗМЕЩЕНИЯ БЕЗ ПОВТОРЕНИЙ. Для поиска максимальной суммы расстояний применим типовой алгоритм ПОИСКА МАКСИМАЛЬНОГО ЭЛЕМЕНТА.

Программа на Бейсике:

```
input "введите количество точек"; n
dim x (n), y (n)
for i=1 to n
  input "введите пары координат"; x(i), y(i)
next
for i=1 to n
  s=0
  for j:=1 to n
   s=s+ sqr ((x(i)-x(j))^2+(y(i)-y(j))^2)
  next
   if s>max then max=s: num=i
next
print "это точка с координатами-"; x(num), y(num)
```

Программа на Паскале:

```
var x,y:array [1..10] of integer;
  i,j,n,num,max:integer;
begin
  writeln ('введите количество точек');
  readln (n);
  for i:=1 to n do
    begin
    writeln ('введите пары координат');
    readln (x[i], y[i]);
    end;
  for i:=1 to n do
        begin
        s:=0;
        for j:=1 to n do
        s:=s+ sqrt (sqr(x[i]-x[j])+sqr(y[i]-y[j]));
        if s>max then
          begin
          max:=s;
          num:=i;
          end;
          end;
  writeln (x[num], y[num]);
End.
```

Тест:

Дано:	N=5 1,6 4,2 6,5 7,10 10,3
Результат:	7,10

На плоскости N точек заданы своими координатами. Найти 2 наиболее удаленные друг от друга точки.

Идея решения: Необходимо применить типовой алгоритм формирования групп СОЧЕТАНИЯ БЕЗ ПОВТОРЕНИЙ. Для поиска двух наиболее удаленных точек применим типовой алгоритм ПОИСКА МАКСИМАЛЬНОГО ЭЛЕМЕНТА.

Программа на Бейсике:

```
input "введите количество точек"; n
dim x(n), y(n)
for i = 1 to n
  input "введите пары координат"; x(i), y(i)
next
for i = 1 to n - 1
  for j = i + 1 to n
   ras = sqr((x(i)-x(j))^ + (y(i)-y(j))^2)
   if ras < max then max = ras: num1 = i: num2 = j
next j, i
print x(num1); y(num1); "-"; x(num2); y(num2)
```

Программа на Паскале:

```
var x,y:array [1..10] of integer;
  n,i,j,num1,num2:integer;
  ras,max:real;
begin
  writeln ('введите количестов точек');
  readln (n);
  for i:=1 to n do
    begin
        writeln ('введите пары координат');
        readln (x[i], y[i]);
        end;
  {================================}
   for i:=1 to n-1   do
        for j:=i+1 to n  do
          begin
          ras:= sqrt(sqr(x[i] - x[j]) +sqr (y[i] - y[j])) ;
          if ras > max then
           begin
       max:=ras;
           num1:=i;
       num2:=j;
           end;
           end;
writeln (x[num1], y[num1], '-', x[num2], y[num2]);
end.
```

Тест:

Дано:	N=6 3,5 6,8 7,6 9,12 15,10 13,3
Результат:	3,5 - 15,10

На плоскости N точек заданы своими координатами. Найти минимальный радиус окружности, включающей в себя все точки.

Идея решения:

Минимальной будет окружность, на которой находятся хотя бы три точки (рис.11.2). Необходимо найти такие три точки, сумма расстояний между которыми максимальна. Необходимо применить типовой алгоритм формирования групп СОЧЕТАНИЯ БЕЗ ПОВТОРЕНИЙ.

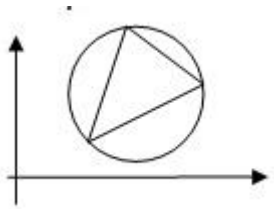

Рис. 11.2.

Радиус описанной окружности:R=(abc)/(4S)

В приведенных ниже программах находятся координаты трех, наиболее удаленных друг от друга точек. Вычислить R не составит труда (проделайте это самостоятельно).

Программа на Бейсике:

```
input "n="; n
dim x(n), y(n)
print "x=, y="
for i = 1 to n
  input x(i), y(i)
next
for i = 1 to n - 2
  for j = i + 1 to n - 1
    for r = j + 1 to n
          ras1 = sqr((x(r) - x(j)) ^ 2 + (y(r) - y(j)) ^ 2)
          ras2 = sqr((x(j) - x(i)) ^ 2 + (y(j) - y(i)) ^ 2)
          ras3 = sqr((x(i) - x(r)) ^ 2 + (y(i) - y(r)) ^ 2)
          if (ras1+ras2+ras3)>max then max=(ras1+ras2+ras3): num1=i: num2=j:
num3=r
next r, j, i
print x(num1); y(num1); "-"; x(num2); y(num2); "-"; x(num3); y(num3)
```

Программа на Паскале:

```
var x,y: array [1..10] of integer;
  n,i,j,r,num1,num2,num3: integer;
  ras1,ras2,ras3,max: real;
begin
  writeln ('введите количестов точек'); readln (n);
  for i:=1 to n do
    begin
        writeln ('введите пары координат'); readln (x[i], y[i]);
        end;
  max:=0;
  for i:= 1 to (n - 2) do
    for j:= i + 1 to (n - 1)  do
          for r:= j + 1 to n do
```

```
          begin
          ras1:=sqrt(sqr(x[r]-x[j])+sqr(y[r]-y[j]));
          ras2:=sqrt(sqr(x[j]-x[i])+sqr(y[j]-y[i]));
          ras3:=sqrt(sqr(x[i]-x[r])+sqr(y[i]-y[r]));
          if (ras1+ras2+ras3)>max then
            begin
                max:=(ras1+ras2+ras3);         num1:=i; num2:=j; num3:=r;
                 end;
                end;
  writeln (x[num1],' ',y[num1], '-',x[num2],' ',y[num2], '-', x[num3],'
',y[num3]);
end.
```

Тест:

Дано:	N=8 4,11 3,7 4,3 7,10 6,7 6,2 11,8 9,5
Результат:	4,11-6,2-11,8

Ключевые термины

- Комбинаторная группа - выборка элементов из исходного множества.
- Размещения - формирование комбинаторных групп по правилам: считать разными выборки, в которых один и тот же элемент занимает разные позиции. Существуют с повторениями и без.
- Сочетания - считать одинаковыми выборки, в которых один и тот же элемент занимает разные позиции. Существуют с повторениями и без.
- Перестановки - в выборке участвуют все элементы исходного множества (K=N). Перестановки с повторениями возможны, когда в исходном множестве есть повторяющиеся элементы
- (либо не считать), допускать повторение одного и того же элемента в выбираемой группе (либо не допускать) и др.

Краткие итоги

Примеры формирования комбинаторных групп из исходного множества {0, 1, 2} по 2 приведены в таблице 11.2:

Таблица 11.2.

Размещения		**Сочетания**	
С повторениями	**Без повторений**	**С повторениями**	**Без повторений**
00, 01, 02, 10, 11, 12, 20, 21, 22	01, 02, 10, 12, 20, 21	00, 01, 02, 11, 12, 22	01, 02, 12
Группы {01} и {10} считаются различными	Исключаются группы, в кот. один и тот же элемент стоит в разных	Группы {01} и {10} считаются одинаковыми	Исключаются группы, в кот. один и тот же элемент стоит в разных

	позициях		позициях

В перестановках участвуют все элементы исходного множества (K=N). Перестановки с повторениями возможны, когда в исходном множестве есть повторяющиеся элементы.

Набор для практики

Вопросы.

- Назовите основные типы комбинаторных групп.
- По каким правилам создаются всевозможные комбинации предметов определенного типа из набора данных?
- В каких случаях возможно сформировать перестановки с повторениями?

Упражнения.

- На карте обозначены N станций (названия станций - A1, A2, A3, …). Требуется рассчитать материальные затраты на строительство автомобильных дорог между станциями. Для этого необходимо написать программу, выдающую запрос на ввод с клавиатуры стоимости дороги между двумя всевозможными станциями и выдающую на экран суммарную стоимость затрат на строительство.
- В некоторой фирме составляют график отпусков. Каждый сотрудник уходит 2 раза в год в 2-хнедельный отпуск. Написать программу для вывода на экран табличной ведомости для заполнения графика отпусков с 2 колонками: первой в формате "Месяц-1_Месяц 2" и второй пустой (в ней сотрудники будут ставить свои ФИО).
- При помощи 5 различных трафаретов нужно нарисовать три картинки в ряд. Выведите на экран всевозможные комбинации.
- На 4 уроках в начальной школе преподают предметы: математику, письмо, чтение, рисование, музыку, физкультуру и труд. Вывести на экран всевозможные варианты расписания предметов на день.

12. Лекция: Комбинаторика. Формирование комбинаторных групп из N по K (K - от 1 до N)

В лекции рассмотрен второй класс задач, в которых N ОПРЕДЕЛЕНО, а K НЕ ОПРЕДЕЛЕНО (N - количество предметов в исходном множестве, K - количество предметов, выбираемых из исходного множества). **Цель лекции:** научиться создавать комбинаторные группы двоичным перебором.

Пусть N=4. Необходимо сформировать различные группы элементов выбираемых из исходного множества. Количество элементов в выборке от 1 до N. Элементы исходного множества будем хранить в массиве A (рис. 12.1):

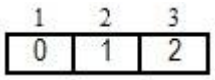

Рис. 12.1.

Составим таблицу, в которой выбранный элемент отметим "1", невыбранный - "0":

Таблица 12.1.

1	2	3
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Итого, сформированы группы: {3}, {2}, {2, 3}, {1}, {1,3}, {1, 2}, {1, 2, 3}.

Для формирования групп потребовалось перебрать все варианты комбинаций "0" и "1". Такой метод формирования комбинаторных групп называется "Двоичным перебором", а количество групп будет равно 2^{n-1}.

Идея решения: для выбора элементов из исходного множества необходимо получить двоичный код (на единицу больше предыдущего). Первый вариант получения нового двоичного кода - перевод счетчика цикла i из десятичной в двоичную систему счисления. Второй вариант получения очередного двоичного кода - ищем в массиве двоичных кодов d последний нулевой элемент , заменяем его на единицу и обнуляем все следующие за ним элементы (этот метод называется лексигрофическим порядком).

Количество возможных комбинаций двоичных кодов 2^n-1 (исключаем двоичный код, состоящий из одних нулей).

Программная реализация на Бейсике:

```
input "введите количество элементов исх. множества="; n
for i=1 to n
```

```
  input "введите элемент"; a(i)
next
for i=1 to 2^n-1
  rem=поиск первого нулевого элемента=========
  for j=1 to n
    if d(j)=0 then x=j
  next
  rem=формирование двоичного кода==========
  for z=x to n
    d(z)=0
  next z
  d(x)=1
  rem=печать элементов==================
  for j=1 to n
    if d(j) <> 0 then print a(j);
  next j
  print
next i
```

Программная реализация на Паскале:

```
const nn=10;
var a,d: array [1..nn] of integer;
  i,n,x,j,z,st: integer;
begin
  writeln ('количество элементов');
  readln (n);
  for i:= 1 to n do
    begin
        writeln ('введите элемент');
        readln (a[i]);
        end;
  {=формирование двоичного кода===}
  st:=1;
  for i:=1 to n do
    st:=st*2;
  for i:= 1 to (st-1) do
        begin
        for j:= 1 to n do
          if d[j]= 0 then x:= j;
        for z:= x to n do
          d[z]:=0;
        d[x]:=1;
        {=печать элементов========}
        for j:= 1 to n do
          if d[j]<>0 then write (a[j]);
        writeln;
        end;
end.
```

Тест:

Дано:	N=3 {1,2,3}
Результат:	3 2 2,3

	1
	1,3
	1,2
	1,2,3

Если предположить, что каждый элемент из исходного набора может повторяться во вновь созданной комбинаторной группе от 0 до n раз, то необходимо организовать n-ричный перебор.

Задачи:

- "Размен монет": дана купюра достоинством X. Требуется разменять ее монетами по 1, 5, 10, 50 рублей.
- Даны гири массами M1, M2, M3, M4. Как можно взвесить предмет массой X?
- Даны N чисел. Выделите из них группы, содержащие от 1 до N элементов, каждая из которых имеет сумму X.

Программная реализация на Бейсике:

```
input "x="; x
input "количество элементов в исходном множестве"; n
for i = 1 to n
  input "введите элемент"; a(i)
next
for i = 1 to (2^n - 1)
  rem==получение следующего двоичного кода==
  for j = 1 to n
    if d(j) = 0 then k = j
  next j
  for z = k to n
    d(z) = 0
  next z
  d(k) = 1
  rem============================
  s = 0
  for j = 1 to n
    if d(j) <> 0 then s = s + a(j)
  next j
  rem========вывод результата==========
  if s = x then
    for ii = 1 to n
          if d(ii) <> 0 then print "  "; a(ii);
       next ii
  end if
  print
next i
```

Программная реализация на Паскале:

```
const nn=10;
var a,d: array [1..nn] of integer;
  ii,i,n,x,j,z,st,k: integer;
begin
  writeln ('введите с чем сравнивать);
  readln (x);
```

```
  writeln ('введите количество элементов');
  readln (n);
  for i:= 1 to n do
    begin
        writeln ('введите элемент');
        readln (a[i]);
        end;
  {=вычисление количества возможных комбинаций=}
  st:=1;
  for i:=1 to n do
    st:=st*2;
  {================================}
  for i:= 1 to (st-1) do
    begin
        {=получение следующего двоичного кода===}
        for j:= 1 to n do
          if d[j]= 0 then k:= j;
        for z:= k to n do
          d[z]:=0;
        d[k]:=1;
        {============================}
        s: = 0;
        for j: = 1 to n do
          if d[j] <> 0 then s:= s + a[j];
        if s = x then
          {=====вывод результата=========}
          for ii:= 1 to n do
            if d[ii] <> 0 then write (a[ii]);
        writeln;
        end;
end.
```

Тест:

Дано:	x=5 n=3 1,2,3
Результат:	2,3

Ключевые термины

- Двоичный перебор - метод формирования комбинаторных групп из исходного множества элементов, на которые "указывают" единицы в соответствующем разряде двоичного кода.
- Лексиграфический порядок - метод получения очередного двоичного кода.

Краткие итоги

Создание комбинаторных групп двоичным перебором основывается на использовании натурального ряда двоичных чисел.

Двоичный код (поразрядно) хранится в дополнительном массиве, имеющем такую же размерность, что и массив с исходным множеством элементов. В массиве двоичных кодов на каждом шаге получаем новый двоичный код, единицы которого "указывают" на соответствующие разряды массива исходного множества.

Получение очередного двоичного кода в лексиграфическом порядке предполагает такой алгоритм: массив с двоичным кодом обходится справа налево, ищется первый ноль, он заменяется на единицу, а все элементы, стоящие левее - обнуляются.

Набор для практики

Вопросы.

- В чем заключается метод двоичного перебора при формировании комбинаторных групп?
- Укажите количество разнообразных n-разрядных двоичных кодов.
- Каким образом можно получить следующее за текущим двоичное число (алгоритм получения)?
- Что изменится в алгоритме формирования комбинаторных групп, если состояние исходных объектов можно охарактеризовать не только как "выбран"/"не выбран": выбранный объект также градируется ("выбран в соответствии с условием 1"/" выбран в соответствии с условием 2")?

Упражнения.

- Ввести с клавиатуры целое число n. Вывести натуральный ряд двоичных чисел до числа, десятичное представление которого не превосходит n.
- Тур-фирма предлагает разнообразные путевки, хранящиеся в базе данных в виде названий туров и их стоимостей (всего n туров). Сделать выборку из базы данных тех туров, которые подходят покупателю по цене (покупатель рассчитывает приобрести не более трех путевок не менее, чем на m рублей).

Дополнительные материалы: Общий глоссарий

- **Палиндром** - запись, читаемая одинаково слева направо и справа налево.
- **Гипотенуза прямоугольного треугольника** - находится по теореме Пифагора как квадратный корень суммы квадратов катетов.
- **Одномерный массив** - именованный набор однотипных переменных, расположенных в памяти непосредственно друг за другом, доступ к которым осуществляется по индексу.
- **Латинский квадрат** - двумерный массив, в каждой строке и столбце которого нет повторяющихся элементов.
- **Циклический сдвиг элементов массива** - сдвиг всех элементов массива вправо (влево) с помещением последнего элемента на первое место (первого элемента на последнее место).
- **Число Армстронга** - сумма цифр этого числа, возведенных в n-ую степень (n - количество цифр числа) равна самому числу.
- **Цифровой корень числа** - получается при сложении цифр числа, затем при сложении цифр вновь полученного числа и так до тех пор, пока в сумме не будет получена одна цифра.
- **Система счисления** - символический метод записи чисел, представление чисел с помощью письменных знаков.
- **"Сверхбольшое число"** - число, величина которого выходит за пределы диапазона допустимых значений выделенной для хранения числа ячейки памяти.
- **Сортировка пузырьком** - один из методов упорядочивания элементов одномерного массива.
- **Двумерный массив** -именованный набор однотипных переменных, расположенных в памяти непосредственно друг за другом, доступ к которым осуществляется по индексу. Массивы с одним индексом называют одномерными, с двумя - двумерными. Квадратный массив - двумерный массив, количество строк и столбцов в котором одинаково.
- **Решето Эратосфена** - метод поиска простых чисел на диапазоне от 1 до N.
- **Арифметический квадрат** - двумерный массив, заполненный так: единичные элементы располагаются в первой строке и первом столбце. Остальные элементы заполняются суммой выше и левее стоящего элемента.
- **Треугольник Паскаля** - образован элементами, расположенными на и выше побочной диагонали арифметического квадрата.
- **Бином Ньютона** - формула для разложения на отдельные слагаемые целой неотрицательной степени суммы двух переменных (формула разложения произвольной натуральной степени двучлена $(a+b)^n$ в многочлен).
- **Путь в двумерном массиве** - набор индексов проходимых элементов, кратчайший путь - набор индексов тех элементов массива, сумма значений которых минимальна
- **Магический квадрат** - двумерный массив, сумма элементов каждой строки, каждого столбца и каждой диагонали которого одинакова.
- **Метод "Террас"** - способ создания магического квадрата
- **Скатерть Улама** - узор из простых чисел, расположенных по спирали.
- **Формула Герона** - формула для вычисления площади треугольника по длинам его сторон.
- **Комбинаторная группа** - выборка элементов из исходного множества.
- **Размещения** - формирование комбинаторных групп по правилам: считать разными выборки, в которых один и тот же элемент занимает разные позиции. Существуют с повторениями и без.

- **Сочетания** - считать одинаковыми выборки, в которых один и тот же элемент занимает разные позиции. Существуют с повторениями и без.
- **Перестановки** - в выборке участвуют все элементы исходного множества (K=N). Перестановки с повторениями возможны, когда в исходном множестве есть повторяющиеся элементы
- **Двоичный перебор** - метод формирования комбинаторных групп из исходного множества элементов, на которые "указывают" единицы в соответствующем разряде двоичного кода.
- **Лексиграфический порядок** - метод получения очередного двоичного кода.