10.09.2020

Семакин и Шестаков – основы алгоритмизации и программирование, к нему +практикум

Алгоритм – описание последовательности действий, строгое выполнение которых приводит к решению поставленной задачи за конечное число шагов.

Алгоритмизация – процесс разработки алгоритма для решения задачи.

Свойства алгоритма:

* Дискретность – раздельность, прерывность
* Детерминированность – определенность, точность
* Конечность
* Массовость
* Результативность

Блок схема – последовательность блоков или геометрических фигур, каждая их которых обозначает строго определённое действие.

**Системы программирования**

СП – комплекс программ и прочих средств, предназначенных для разработки и эксплуатации программ на конкретном языке программирования для конкретного вида ЭВМ

Язык программирования – это совокупность набора символов системы, правил образования, правил истолкования конструкций из символов, для задания алгоритмов с использованием символов естественного языка.

Транслятор – программа переводчик с языка программирования на язык машинных команд. Транслятор работает под управлением операционной системы, которая обеспечивает связь с его устройствами и блоками ЭВМ.

Существуют: интерпретаторы и компиляторы

Библиотеки подпрограмм содержат машинный код программ, реализующий различные стандартные функции.

Редактор связей – выполняет связывание программы и магнитного кода стандартных функций, находя их в библиотеках. На выходе формирует работоспособное приложение – исполняемый код для конкретной задачи.

**Delphi**

**Массивы**

**program** ff;

**Const** n=10;

**type** massiv=**array** [1..n] **of** integer;

**var** a:massiv; i:integer; S:integer;

**begin**

**for** i:=1 **to** n **do**

**begin**

s:=0;

write('a[',i,']=');

readln(a[i]);

**end**;

**for** i:=1 **to** n **do** s:=s+a[i];

writeln(s);

**end**.

**Формирование массива случайным образом**

Тут мы выставляем рандомные оценки ученикам

Оценки должны быть от 2 до 5, тобишь всего 4 разных цифры. Рандом работает только от 0, тоесть над взять random (4)+2. Тоесть было 0,1,2,3; а стало 2,3,4,5.

**program** ff;

**Const** n=25;

**type** massiv=**array** [1..n] **of** integer;

**var** a:massiv; i:integer; S:integer;

**begin**

randomize;

**for** i:=1 **to** n **do**

**begin**

a[i]:=random(4)+2;

writeln('a[',i,']=',a[i]);

**end**;

writeln;

s:=0;

**for** i:=1 **to** n **do** s:=s+a[i];

writeln(s);

**end**.

**Издеваемся над массивами**

Чтобы найти max в массиве, нужно взять первый элемент массива, и его сравнивать с последующим элементом

**program** ff;

**Const** n=10;

**type** massiv=**array** [1..n] **of** integer;

**var** a:massiv; i:integer; max:integer;

**begin**

randomize;

**for** i:=1 **to** n **do**

**begin**

a[i]:=random(100)-50;

writeln('a[',i,']=',a[i]);

**end**;

writeln;

max:=a[1];

**for** i:=2 **to** n **do**

**if** a[i]>max **then** max:=a[i];

writeln('max=',max);

В массиве определить максимальный элемент, минимальный элемент и поменять их местами.

Минимум и максимум имеют начальное значение первого элемента. Также нам надо знать их индексы (i). В начале у обоих будет индекс 1.

**program** ff;

**Const** n=25;

**type** massiv=**array** [1..n] **of** integer;

**var** a:massiv; i:integer; max,min,imax,imin,t:integer;

**begin**

randomize;

**for** i:=1 **to** n **do**

**begin**

a[i]:=random(100)-50;

writeln('a[',i,']=',a[i]);

**end**;

writeln;

max:=a[1]; min:=a[1]; imax:=1; imin:=1;

**for** i:=2 **to** n **do**

**if** a[i]>max **then**

**begin**

max:=a[i];

imax:=i;

**end**

**else if** a[i]<min **then**

**begin**

min:=a[i];

imin:=i;

**end**;

t:=a[imax];

a[imax]:=a[imin];

a[imin]:=t;

writeln(a[imax]);

writeln(a[imin]);

writeln;

**for** i:=1 **to** n **do** writeln('a[',i,']=',a[i]);

**end**.

В одномерном числовом массиве элементы массива переставить в обратном порядке.

For i:=1 to (n div 2) –тип если элементы нечетное кол-во, чтобы последний не трогало, так как он и так в середине стоит.

I+x=n+1

X=n+1-i

**program** ff;

**Const** n=25;

**type** massiv=**array** [1..n] **of** integer;

**var** a:massiv; i:integer; t:integer;

**begin**

randomize;

**for** i:=1 **to** n **do**

**begin**

a[i]:=random(100)-50;

writeln('a[',i,']=',a[i]);

**end**;

writeln;

**for** i:=1 **to** (n **div** 2) **do**

**begin**

t:=a[i];

a[i]:=a[n+1-i];

a[n+1-i]:=t;

**end**;

**for** i:=1 **to** n **do** writeln('a[',i,']=',a[i]);

**end**.

В итоге мы перевернули наш массивчик.

**Двумерные массивы**

Ровный массив 4 на 5

**program** f;

**const** n=4; m=5;

**type**

mas=**array**[1..n,1..m] **of** integer;

**var** a:mas; i,j:integer;

**begin**

randomize;

**for** i:=1 **to** n **do**

**begin**

**for** j:=1 **to** m **do**

**begin**

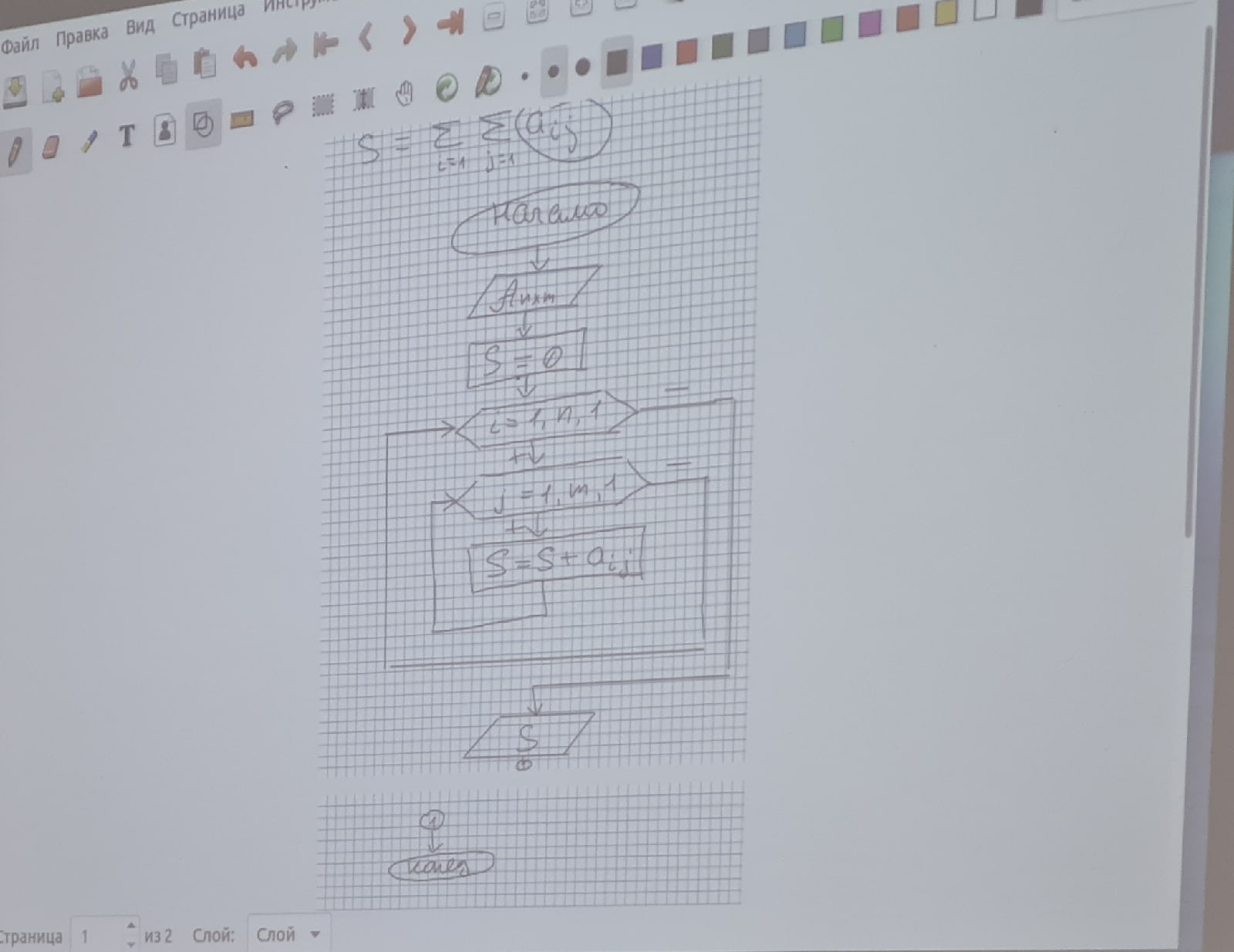
a[i,j]:=random(100)-50;

write(a[i,j]:4);

**end**;

writeln;

**end**;

**end**.

Ровный массив 2 на 2 + сумма всех чисел.

**program** f;

**const** n=2; m=2;

**type**

mas=**array**[1..n,1..m] **of** integer;

**var** a:mas; i,j,s:integer;

**begin**

s:=0;

randomize;

**for** i:=1 **to** n **do**

**begin**

**for** j:=1 **to** m **do**

**begin**

a[i,j]:=random(100)-50;

write(a[i,j]:4);

s:=s+a[i,j];

**end**;

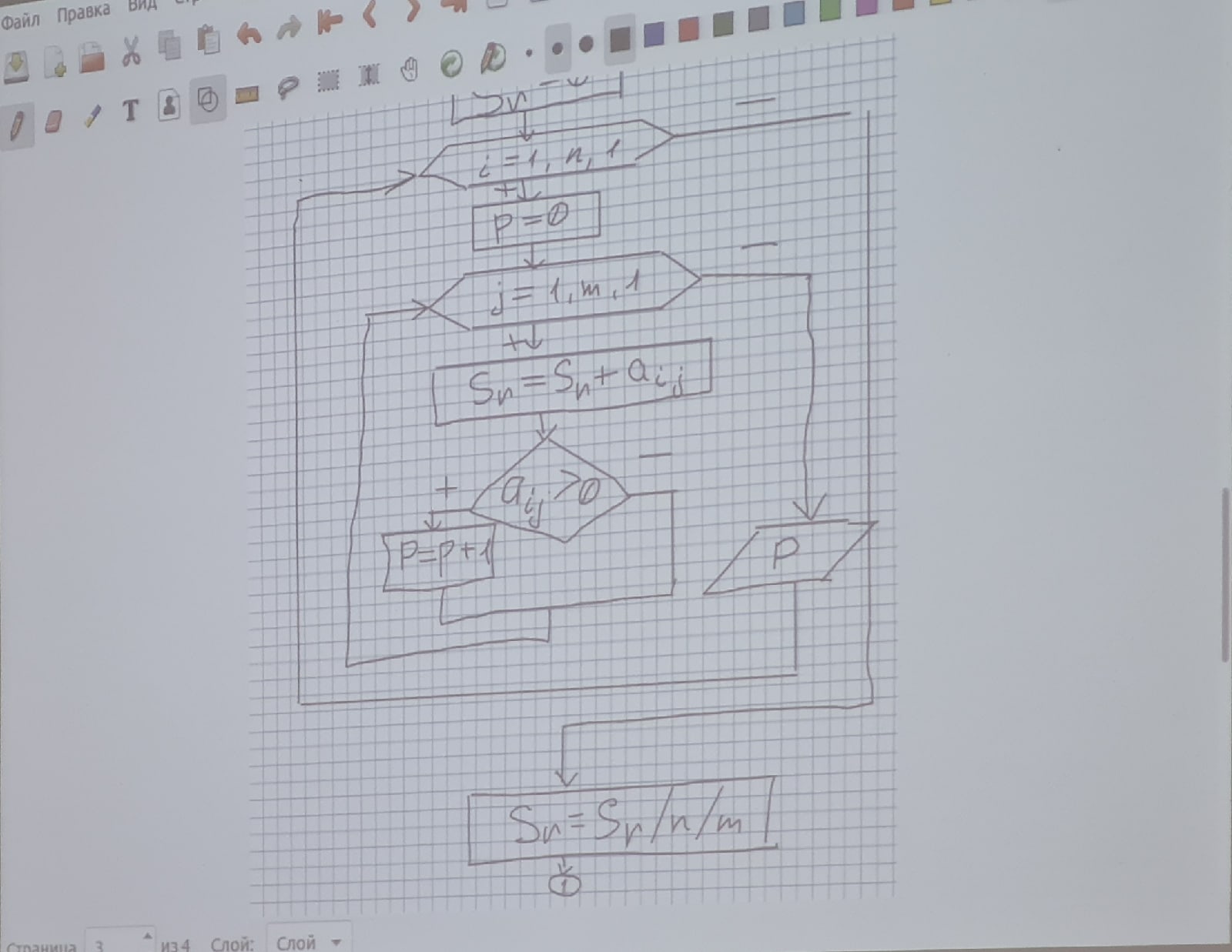
writeln;

**end**;

writeln(s);

**end**.

Найти для всего массива среднее арифметическое всех элементов. И вывести в каждой строке кол-во положительных элементов.

**program** f;

**const** n=4; m=5;

**type**

mas=**array**[1..n,1..m] **of** integer;

**var** a:mas; i,j,k:integer; s:real;

**begin**

s:=0;

randomize;

**for** i:=1 **to** n **do**

**begin**

k:=0;

**for** j:=1 **to** m **do**

**begin**

a[i,j]:=random(100)-50;

write(a[i,j]:4);

s:=s+a[i,j];

**if** a[i,j]>=0 **then** k:=k+1;

**end**;

write(' :elementov', k:2);

writeln;

**end**;

s:=s/(n\*m);

writeln(s);

**end**.

Определить самый первый столбец, который содержит только положительные элементы.

**program** f;

**const** n=4; m=5;

**type**

mas=**array**[1..n,1..m] **of** integer;

**var** a:mas; i,j,num:integer; all:boolean;

**begin**

randomize;

**for** i:=1 **to** n **do**

**begin**

**for** j:=1 **to** m **do**

**begin**

a[i,j]:=random(100)-50;

write(a[i,j]:4);

**end**;

writeln;

**end**;

num:=0;

**for** j:=1 **to** m **do**

**begin**

all:=true;

**for** i:=1 **to** n **do if** a[i,j]<0 **then**

**begin**

all:=false;

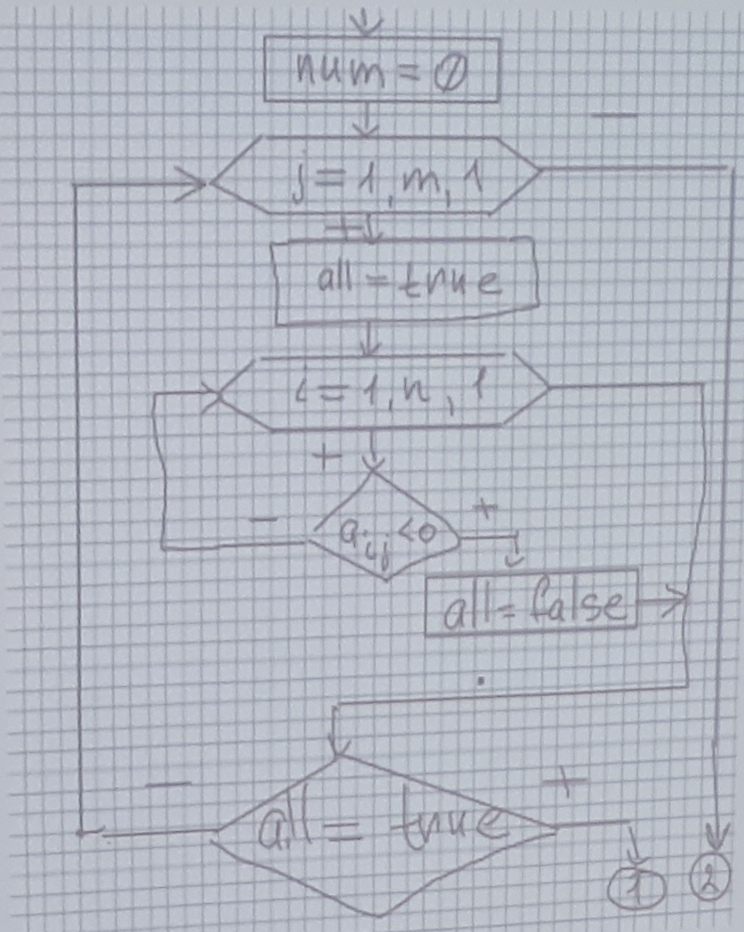
**break**;

**end**;

**if** all **then**

**begin**

num:=j;

 **break**;

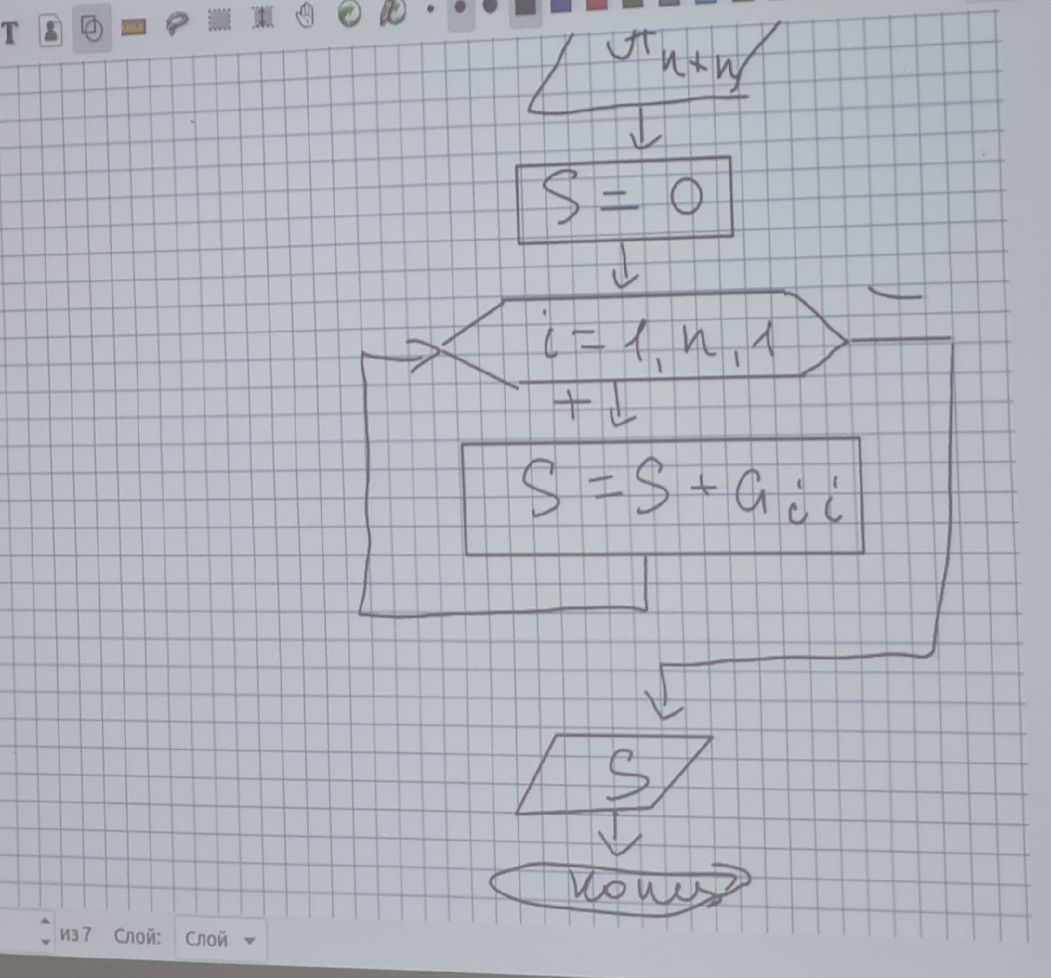
**end**;

**end**;

**if** num=0 **then** writeln ('net')

**else** writeln(num);

**end**.

Посчитать сумму центральной диагонали

**program** f;

**const** n=4; m=4;

**type**

mas=**array**[1..n,1..m] **of** integer;

**var** a:mas; i,j,s:integer;

**begin**

randomize;

**for** i:=1 **to** n **do**

**begin**

**for** j:=1 **to** m **do**

**begin**

a[i,j]:=random(100)-50;

write(a[i,j]:4);

**end**;

writeln;

**end**;

s:=0;

**for** i:=1 **to** n **do** s:=s+a[i,i];

writeln(s);

**end**.

Посчитать сумму элементов ниже главной диагонали

**program** f;

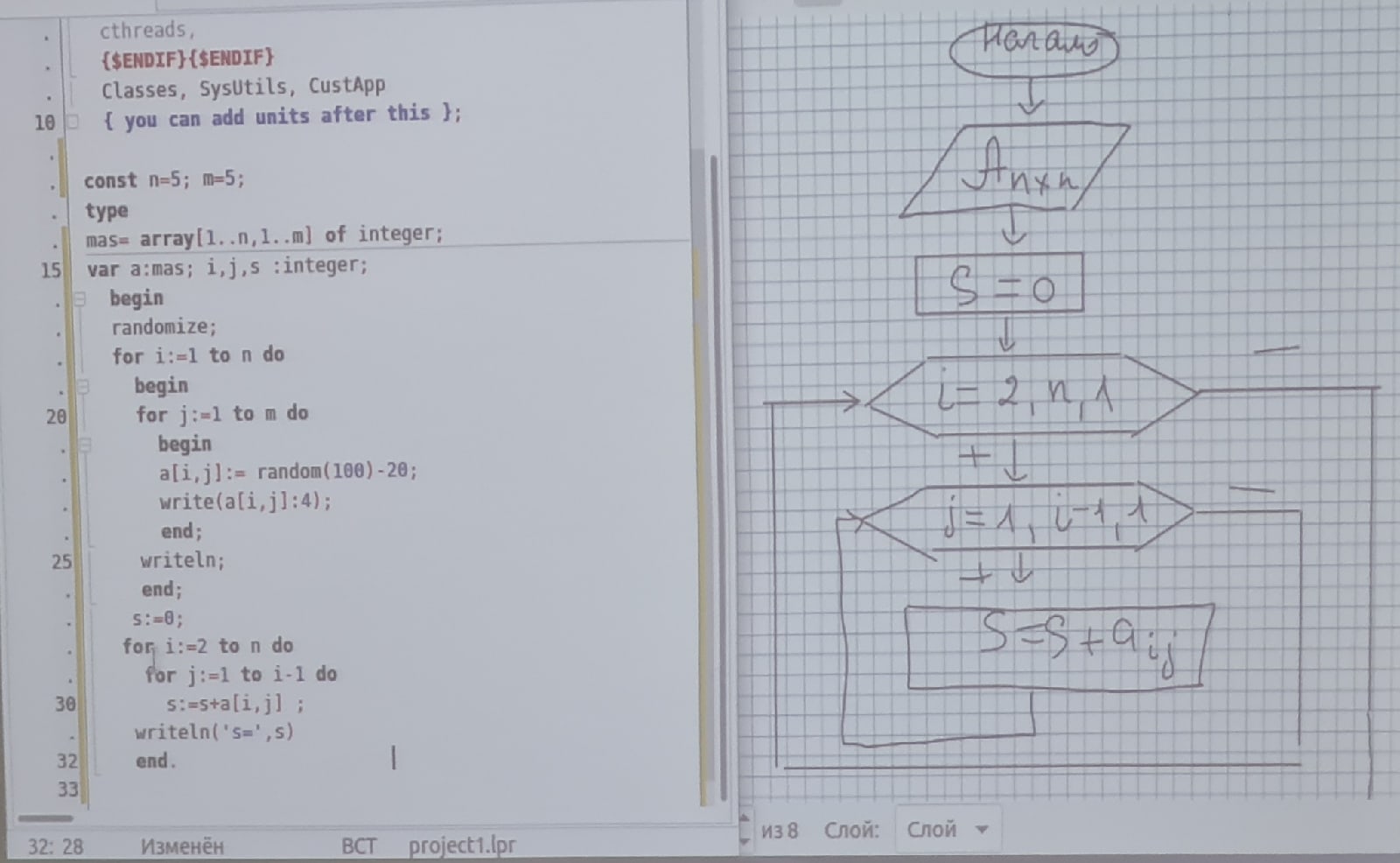
**const** n=4; m=4;

**type**

mas=**array**[1..n,1..m] **of** integer;

**var** a:mas; i,j,s:integer;

**begin**

randomize;

**for** i:=1 **to** n **do**

**begin**

**for** j:=1 **to** m **do**

**begin**

a[i,j]:=random(100)-50;

write(a[i,j]:4);

**end**;

writeln;

**end**;

s:=0;

**for** i:=2 **to** n **do**

**for** j:=1 **to** i-1 **do**

s:=s+a[i,j];

writeln(s);

**end**.

Поставить в строке элементы больший в лево, меньший в право.

**program** f;

**const** n=4; m=4;

**type**

mas=**array**[1..n,1..m] **of** integer;

**var** a:mas; i,j,t, nmax, nmin:integer;

**begin**

randomize;

**for** i:=1 **to** n **do**

**begin**

**for** j:=1 **to** m **do**

**begin**

a[i,j]:=random(100)-50;

write(a[i,j]:4);

**end**;

writeln;

**end**;

**for** i:=1 **to** n **do**

**begin**

nmax:=1;

nmin:=1;

**for** j:=2 **to** m **do**

**if** a[i,j]>a[i,nmax] **then** nmax:=j

**else if** a[i,j]<a[i,nmin] **then** nmin:=j;

t:=a[i,nmax];

a[i,nmax]:=a[i,1];

a[i,1]:=t;

t:=a[i,nmin];

a[i,nmin]:=a[i,n];

a[i,n]:=t;

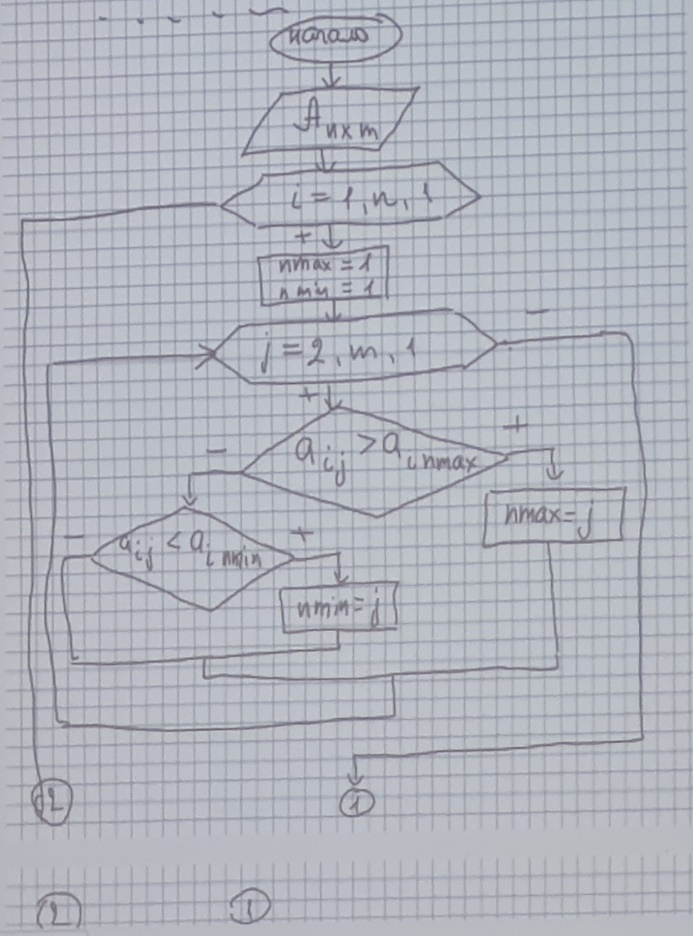
**end**;

writeln;

**for** i:=1 **to** n **do**

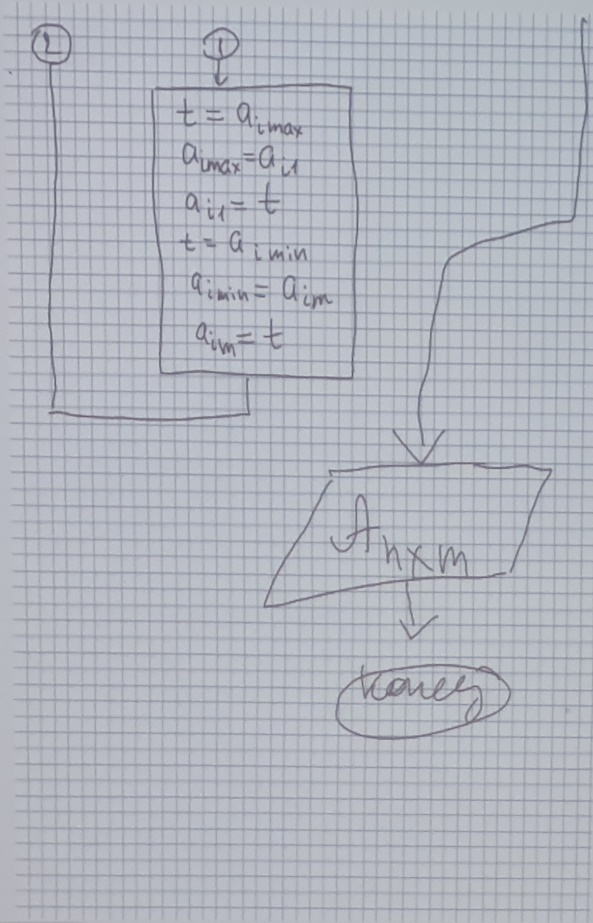
**begin**

**for** j:=1 **to** m **do** write(a[i,j]:4);

 writeln;

**end**;

**end**.



Перемешать элементы в массиве в рандномном порядке.

**program** f;

**const** n=4; m=5;

**type**

mas=**array**[1..n,1..m] **of** integer;

**var** a:mas; i,j,k,rand\_i,rand\_j:integer; all, flag:boolean;

**begin**

randomize;

**for** i:=1 **to** n **do**

**begin**

**for** j:=1 **to** m **do**

**begin**

a[i,j]:=random(100);

write(a[i,j]:4);

**end**;

writeln;

**end**;

**for** i:=1 **to** n **do**

**for** j:=1 **to** m **do**

**begin**

k:=a[i,j];

rand\_i:= random(1, n);

rand\_j:= random(1,m);

a[i,j]:= a[rand\_i, rand\_j];

a[rand\_i, rand\_j]:=k;

**end**;

writeln;

**for** i:=1 **to** n **do**

**begin**

**for** j:=1 **to** m **do**

**begin**

write(a[i,j]:4);

**end**;

writeln;

**end**;

**end**.

**Множественный тип данных**

Type <имя типа> = set of <базовый тип>;

Базовый тип:

* Перечисляемые
* Интервальный
* byte

type

colors = set of (red, green, blue);

caps = set of ‘A’..’Z’;

numbers = set of byte;

var oct: set of 0..7;

[‘A’..’D’]

[1,2,6]

[2,3,10..13]

[] – пустое множество

Тип “множество” задает набор всех возможных подмножеств его элементов, включая пустое

Если базовый тип имеет k элементов, то число подмножеств равно 2k.

Переменная типа “множество” содержит одно конкретное значение подмножествео значений.

*Интервальная переменная:*

Var b:1..3; может принимать значения: 1,2,3

*Множественная переменная:*

Var m: set of 1..3; может принимать значения:

[],[1],[2],[3],[1,2],[1,3],[2,3],[1,2,3]

**type**

digitchar=**set of** '0'..'9';

digit=**set of** 0..0;

**var**

s1,s2,s3: digitchar;

s4,s5,s6: digit;

**begin**

s1:=['1','2','3'];

s2:=['3','2','1'];

s3:=['2','3'];

s4:=[0..3,6];

s5:=[4,5];

s6:=[3..9];

**end**.

**Операции над множествами**

*Пересечение множеств (\*);*

Результат содержит элементы, общие для обоих множеств

s4\*s6 содержит [3,6]

s4\*s5 – пустое множество

*Объединение множеств (+);*

Результат содержит элементы первого множества, дополненные недостающими элементами из второго множества

s4+s5 содержит [0,1,2,3,4,5,6];

s5+s6 содержит [3,4,5,6,7,8,9];

*Разность (-);*

Результат содержит элементы из первого множества, которые не принадлежат второму

s6-s5 содержит [3,6,7,8,9];

s4-s5 содержит [0,1,2,3,6];

Проверка эквивалентности (=); возвращает True, если оба множества эквивалентны.

Проверка неэквивалентности (<>); возвращает True, если оба множества неэквивалентны.

Проверка вхождения (<=); возвращает True, если первое множество включено во второе;

Проверка вхождения (>=); возвращает True, если второе множество включено в первое;

Проверка принадлежности (in); в этой бинарной операции первый элемент – выражение, а второй – множество одного и того же типа; возвращается True, если выражение имеет значение, принадлежащее множеству.

Задано множество, основанное на значения прописных латинских букв.

**type** caps=**set of** 'A'..'Z';

**var** a,b,c:caps;

**begin**

a:=['A','U'..'Z'];

b:=['M'..'Z'];

c:=a+b;

writeln(c);

c:=a\*b;

writeln(c);

c:=b-a;

writeln(c);

c:=a-b;

writeln(c);

**if** a=b **then** writeln('oaoao');

**if** a<>b **then** writeln('aoaoa');

**if** c<=a **then** writeln('cacacac');

**if** 'N' **in** b **then** writeln('NNNN');

**end**.

В автомобильных гонках учувствуют 30 машин со стартовыми номерами от 101 до 130. Проводится 5 заездов по 6 машин. Сформировать состав заездов случайным образом.

**type**

s=**set of** 101..130;

**var** sor, za:s; i, j, n: integer;

**begin**

randomize;

sor:=[];

**for** i:=1 **to** 5 **do begin** za:=[];

**for** j:=1 **to** 6 **do**

**begin**

**repeat**

n:=random(30)+101;

**until not**(n **in** za) **and not**(n **in** sor);

za:=za+[n];

**end**;

sor:=sor + za;

writeln(za);

**end**;

**end**.

**type**

s=**set of** 101..130;

**var** sor, za:s; i, j, n: integer;

**begin**

randomize;

sor:=[];

**for** i:=1 **to** 5 **do begin** za:=[];

write(i, ' zaezd ');

**for** j:=1 **to** 6 **do**

**begin**

**repeat**

n:=random(30)+101;

**until not**(n **in** sor);

za:=za+[n];

write(n:4);

sor:=sor + [n];

**end**;

writeln;

**end**;

**end**.

**type**

s=**set of** 101..130;

**var** sor, za:s; i, j, n: integer;

**begin**

randomize;

sor:=[];

**for** i:=1 **to** 5 **do begin** za:=[];

**for** j:=1 **to** 6 **do**

**begin**

**repeat**

n:=random(30)+101;

**until not**(n **in** sor);

za:=za+[n];

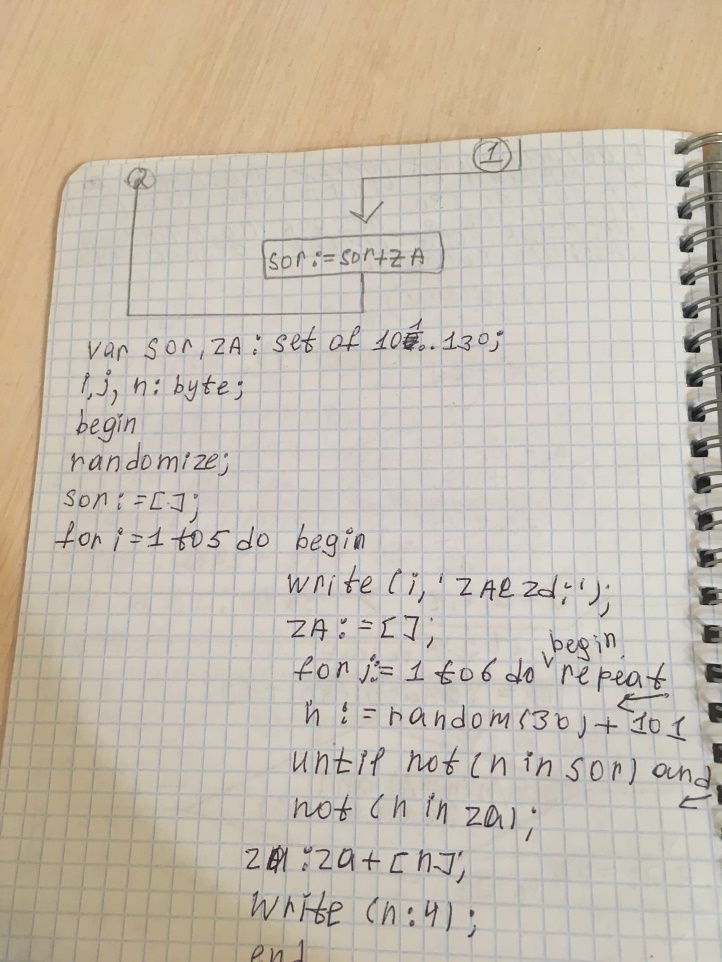
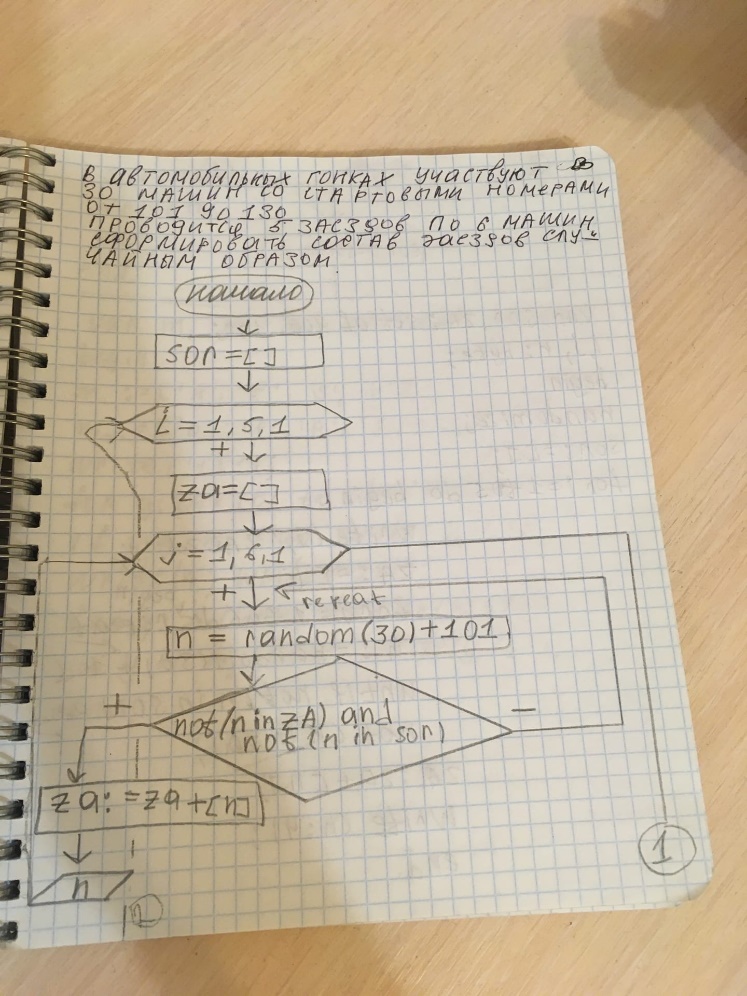
sor:=sor + [n];

**end**;

writeln(za);

**end**;

**end**.



Вывести множество

**var** m1,m2:**set of** char; y:char;

**begin**

m1:=['A','F','S','P'];

m2:=['O','P','R','S','T'];

write('1: ');

**for** y:=chr(0) **to** chr(255) **do**

**if** y **in** m1 **then** write(y,', ');

writeln;

write('2: ');

**for** y:=chr(0) **to** chr(255) **do**

**if** y **in** m2 **then** write(y,', ');

**end**.

**Строки**

Стандартные (string)

Определяемые на основе string

Строки динамической памяти

Строка string может содержать до 255 символов. Под одну переменную типа string всегда отводится 256 байт.

**Присваивание**

S2:=’nrtc’;

S1:=S2;

**Конкатенация**

S1:=’ком’;

S2:=S1+’пот';

**Операции отношения**

‘abc’>’ab’

‘abc=’abc’

‘abc’<’ab ’

**Процедуры ввода/вывода**

Readln(s1,s2);

Write(s1);

**Обращение к элементу**

S1[4]:=’x’;

Writeln(s2[3] + s2[5] + ‘r’);

**Процедуры и функции**

Concat(S1 [, S2, … , Sn]:String):String;

Возвращает строку, представляющую собой сцепление строк-параметров S1, S2,…,Sn

S1:=’ком’;

S2:=’пот’;

S3:=concat(S1,S2);

Copy(st:String; Index, Count:integer): String;

Из строки st копирует Count символов, начиная с символа с номером Index

S1:=’программирование’;

S2:=copy(S1,4,5);

S2:=’грамм’;

Delete(St:string; Index, Count: Intger);

Удаляет Count символов из строки St, начиная с символа с номером Index

S1:’программирование’;

Delete(S1,9,4); S1=’программание’

Delete(S1,10,3); S1=’программа’

Insert(SubSt, St: String; Index: Integer);

Вставляет подстроку SubSt в строку St, начиная с символа с номером Index

S1:'программа’;

Delete(S1,9,1); S1=’Программ’

Insert(‘ист’,S1,9); S1=’Программист’

Length(St: String): Integer;

Возвращает текущую длину строки St

S1:=’программирование’;

X:=length(S1);

X=16

Pos(SubSt, St: String): Integer;

Отыскивает в строке St первое вхождение подстроки SubSt и возвращает номер позиции, с которой она начинается. Если подстрока не найдена, возвращается нуль.

S1:’программирование’;

X:=pos(‘ров’,S1);

X=10

StrToFloat(St: String): Extendet;

Преобразует символы строки St в вещественное число. Строка не должна содержать ведущих или ведомых пробелов

S1:=’45.78’;

X:=strtofloat(S1);

X=45.78

StrTolnt(St: String): Integer;

Преобразует символы строки St в целое число. Строка не должна содержать ведущих или ведомых пробелов

S1:’498’;

X:=strtolnt(S1);

X=498

Val(St: String; var X; Code: Integer);

Преобразует строку символов St во внутреннее представление целой или вещественной переменной x, которой определяется типом этой переменной. Параметр Code содержит ноль, если преобразование прошло успешно, и тогда в X помещается результат преобразования, в противном случае он содержит номер позиции в строке St, где обнаружен ошибочный символ, и в этом случае содержимое X не меняется. В строке St содержит символьное представление вещественного числа, разделителем целой и дробной частей должна быть точка независимого от того, каким символом этот разделитель указан в Windows.

St:=’45.7e8’;

Val(st,x,y);

St:=4570000000

FloatToStr(Value: Extended): String;

Преобразует вещественное значение Value в строку символов

X:=45.78;

S1:=floattostr(x);

S1=’45.78’

IntToStr(Value: Integer): String;

Преобразует целое число Value в строку символов

X:=459;

S1:=inttostr(x);

S1=’459’

Str(X [:Width [:Decimals]]; vat St:String);

Преобразует число X любого вещественного или целого типов в строку символов St; параметры Width и Decimals, если они присутствуют, задают формат преобразования; Width определяет общую ширину поля, выделенного под соответствующее символьное представление числа X, а Decimals – количество символов дробной части (этот параметр имеет смысл только в том случае, когда X – вещественное число)

X:=45.78;

Str(x:6:3,st);

**Задачки**

В строке подсчитать количество букв a и заменить их буквами b

N:=0

**While** pos(‘a’,s)>0 **do**

**Begin**

N:=n+1;

S[pos(‘a’,s)]:=’b’;

**End**;

Определить, является ли введенная строка перевертышем

**For** i:=1 **to** trunc(length(s)/2) **do**

**Begin**

**If** s[i]<>s[length(s)-i+1] **then**

**Begin**

Writeln(‘не перевертыш’);

**Exit**;

**End**;

**End**;

Writeln(‘перевертыш’);

**Тип данных ‘Запись’**

Тип «запись» предназначен для объединения разнотипных данных

Он вводится с помощью ключевого слова **record**

Элементы записи называются полями

**Тип данных «Файл»**

**Файл** – именованная область памяти на внешнем носителе компьютера

Характерные особенности:

* Имя файла
* Компоненты одного типа
* Объем, ограничивается только возможностями компьютера

**Типы файлов:**

* Типизированные файлы <имя> : File of <тип>;
* Текстовые файлы <имя> : Text;
* Нетипизированные файлы <имя> : File;

**type**

product=**record**

name: string;

code: word;

cost: comp;

**end**;

text80=**file of** string[80];

**var**

f1: **file of** char;

f2: textfile;

f3: **file**;

f4: text80;

f5: **file of** product;

**Алгоритм работы с файлами**

1. Объявить файловую переменную

2. Связать ее с физическим файлом

3. Открыть файл для чтения или записи

4. Выполнить операции ввода/вывода

5. Закрыть файл

**Программы для работы с файлами**

Prodecure assign (var F; FileName:String);

Связывает файловую переменную F с именем файла FileName

Procedure close (var F); Закрывает файл

Связь файловой переменной F с именем файла, устанавливается ранее процедурой assign, сохраняется.

Procedure erase (var F);

Уничтожает файл, с которым была ассоциирована переменная F. Перед выполнением процедуры необходимо закрыть файл.

Procedure rename (var F; NewName: String);

Переименовывает файл F; NewName = строковое выражение, содержащее новое имя файла. Перед выполнением процедуры необходимо закрыть файл.

Procedure reset (var F: File);

Открывает существующий файл для чтения

Procedure rewrite (var F: File);

Открывает существующий файл для записи данных. Если файл существовал ранее, он очищается. Если файла не было, он создается.

Function eof (var F): Boolean;

Тестирует конец файла и возвращает True, если файловый указатель стоит в конце файла.

Function fileexists (const FileName: String): Boolean;

Возвращает True, если файл с именем (и, возможно, маршрутом доступа) FileName существует.

Текстовые файлы предназначены для хранения текстовой информации

Текстовые файлы можно открывать не только для чтения или записи, но и для добавления информаци в конец файла:

procedure append (var F);

procedure read (var F: TextFile; V1 [, V2,…, Vn]);

Читает из текстового файла последовательность символьных представлений переменных Vi типа Char, String, а также любого целого или вещественного типа.

Procedure readln (var F: TextFile; [V1 [, V2,…, Vn]]);

Читает из текстового файла последовательность символьных представлений переменных vi типа char, String, а также любого целого или вещественного типа с учетом границ строк.

Procedure write (var F: Text; P1 [, P2,…, Pn]]);

Записывает символьные представления параметров Pi и признак конца строки в текстовый файл.

Function eoln (var F: Text): Boolean;

Пропускает все пробелы, знаки табуляции и признаки конца строки и возвращает True, если признак конца файла обнаружен.

Function sekkeol (var F: TextFile): Boolean;

Пропускает все пробелы и знаки табуляции до признака конца строки и возвращает True, если признак обнаружен.

**Типизированные файлы**

Длина любого компонента типизированного файла строго постоянна, что дает возможность организовать прямой доступ к каждому из них (то есть доступ к компоненту по его порядковому номеру).

Function FilePos (vaf F): LongInt;

Возвращает текущую позицию в файле, то есть номер компонента, которой будет обрабатываться следующей операцией ввода-вывода.

Function FileSize (var F): LongInt;

Возвращает количество компонентов файла.

Procedure Seek (var F; N: LongInt);

Смещает указатель файла F к требуемому компоненту: N – номер компонента файла (первый компонент файла имеет номер 0)

Procedure trancate (var F);

Устанавливает в текущей позиции признак конца фала и удаляет все последующие блоки)

**Указатели**

Указатель – переменная, которая в качестве своего значения содержит адрес байта памяти. С помощью указателей можно размещать в динамической памяти любой из известных в Delphi типов данных.

**type**

pword=^word;

**var**

pw:word;

p1,p2:^integer;

p3:^real;

**type**

perconpointer=^perconpointer;

preconrecord=**record**

name:string;

job:string;

next:parconpointer

**end**;

Можно объявлять указатель и не связывать его при этом с каким-либо конкретным типом данных

**var**

p:pointer;

**Операции с указателями**

Присваивание

p:=nil;

p2:=p1;

**Процедуры и функции**

**procedure new (var p:pointer));** Резервирует фрагмент динамической памяти для размещения переменной и помещает типизированный указатель P адрес первого байта

**var**

p,p1:^integer;

p2:^real;

**begin**

**new**(p);

**new**(p2);

**end**.

Разыменовывание указателя. Значение, которое адресует указатель, то есть собственно данные, размещенные в куче, обозначаются знаком ^ сразу за указателем

pj^ := 2; // в область памяти, на которую указывает указатель

pR^ := 2\*pi;

**function Addr(x): pointer;** Возвращает адрес аргумента x. Аналогичный результат возвращает операция @

**var**

x:word;

pw:^word;

**begin**

pw:=addr(x);

pw:=@(x)

**end**.

**procedure GetMem( var P:pointer; Size: integer);** Резервирует за не типизированным указателем P фрагмент динамической памяти требуемого размера Size

Пример:

**type** rec = **record**

d : word;

s : string;

**end**;

pword =^ word;

**var**

p1, p2 : pword;

p3 :^ rec;

**begin**

**new**(p1);

p2 := **new**(pword);

**new**(p3);

p1^:2;

p2^:4;

p3^.d:=p1^;

p3^.s:='nrtc';

inc(p1^);

p2^:=p1^+p3^.d;

**with** p3^ **do** writeln(d,s);

**end**.

**procedure dispose(var p:pointer);** Возвращает к кучу фрагмент динамической памяти, которым ранее был зарезервирован за типизированным указателем P

**procedure FreeMem(var P:pointer; Size: integer);** Возвращает в кучу фрагмент динамической памяти, который ранее был зарезервирован за не типизированным указателем p

**procedure mark(var p:pointer);** Записывает в указатель p адрес начала участка свободной динамической памяти на момент ее вызова. Вызывается до начала выделения памяти, которую замет потребуется освободить.

**procedure release (var p:pointer);** Освобождает участок динамической памяти, начиная с адреса, записанного в указатель p процедурой mark

**Динамические структуры данных**

**Линейные списки**

В линейном списке каждый элемент связан со следующим и, возможно, с предыдущим. В первом случае список называется односвязным, во втором двусвязным

Если последний элемент связать с указателем с первым, получится кольцевой список.

**Операции со списками**

Формирование односвязного списка, вставка и удаление его элементов

**program** linked\_list;

**const** n=5;

**type** pnode=^node;

node=**record**

d: word;

s: string;

p: pnode;

**end**;

**var** beg: pnode;

i, key: word;

s: string;

option: word;

**const** text: **array**[1..n] **of** string = ('one','two','three','four','five');

Добавление элемента в список

**procedure** add (**var** beg:pnode; d:word; cons s:string);

**var**

p:pnode; //указатель на создаваемый элемент

t:pnode; //указатель для просмотра списка

**begin**

**new**(p); //создание элемента

p^.d:=d;

p^.s:=s; //заполнение элемента

p^.p:=nil;

**if** beg=nil **then** beg:=p //список был пуст

**else begin**

t:=beg;

**while** t^.p<>nil **do** //проход по списку до конца

t:=t^.p;

t^.p:=p; //привязка нового элемента к последнему

**end**;

**end**.

Поиск элемента по ключу

**function** find (beg: pnode; key:word; **var** p,pp:pnode): boolean;

**begin**

p:=beg;

**while** p<>nil **do begin**

**if** p^.d=key **then begin**

find:=true;

**exit**

**end**;

pp:=p;

p:=p^.p;

**end**;

find:=false;

**end**;

Вставка элемента

**procedure** insert(beg:pnode; key,d:word; **const** s:string);

**var**

p:pnode;

pkey:pnode;

pp:pnode;

**begin**

**if not** fing (beg, key, pkey, pp) **then begin**

writeln('Вставка не выполнена');

**exit**

**end**;

**new**(p);

p^.d:=d;

p^.s:=s;

p^.p=pkey^.p;

pkey^.p:=p;

**end**.

Удаление элемента

**procedure** del (**var** beg: pnode; key:word);

**var**

p: pnode; //указатель на удаляемый элемент

pp: pnnode;

**begin**

**if not** find (beg, key, p, pp) **then begin**

writeln('Удаление не выполнено');

**exit**;

**end**;

**if** p=beg **then** beg:=beg^.p // удаление первого элемента

**else** pp^.p:=p^.p;

dispose(p);

**end**.

Вывод списка

**procedure** print(beg=pnode);

**var**

p:pnode;

**begin**

p:=beg;

**while** p<>nil **do begin**

writeln(p^d:3, p^.s);

p:=p^.p

**end**;

**end**.

Главная программа

**begin**

**for** i:=1 **to** 5 add (beg, i, text[i]);

**while** true **do begin**

writeln('1-вставка, 2-удаление,3-вывод, 4-выход);

readln(option);

**case** option **of**

1: **begin**

writeln('ключ для вставки');

readln(key);

writeln('Вставленный элемент');

readln(i);

readln(s);

**end**;

2: **begin**

readnl(key);

del(beg, key);

**end**;

3: **begin**

print(beg);

**end**;

4: **exit**;

**end**;

**end**;

**end**.