**03.06.20**

Уважаемые студенты группы ИС21. Сегодня мы снова изучаем теорию. Вам необходимо будет выполнить следующие задания: внимательно изучите материал урока; для подготовки к экзамену и практической работы в тетрадь напишите конспект. На следующем занятии у вас будет практическая работа по тем вариантам, которые вам были определены перед первой практической работой.

Отчет о работе отправьте его по электронной почте на yun707@yandex.ru. При отправлении выполненных заданий укажите ФАМИЛИЮ СТУДЕНТА, в Теме НАИМЕНОВАНИЕ ДИСЦИПЛИНЫ и НАЗВАНИЕ ВЫПОЛНЕННОЙ РАБОТЫ.

**Тема: Понятие подпрограмм, подпрограммы – процедуры и подпрограммы - функции. Описание процедур. Выполнение процедур. Описание функций, их выполнение. Виды параметров в подпрограммах.**

Процедуры и функции являются важным инструментом большинства языков программирования. С их помощью можно часто встречающиеся фрагменты программы, одинаковые по выполняемым действиям, но различающиеся только значениями исходных данных, скомпоновать в группу операторов. Для эффективного программирования подобных повторений фрагментов в языках программирования введено понятие подпрограммы.

***Подпрограмма*** – это группа операторов, оформленная как самостоятельная программная единица.

Подпрограмма записывается однократно в определенной части программы, а затем в нужных местах программы обеспечивается обращение к ней. При обращении к подпрограмме ей передаются исходные данные, а после выполнения операторов подпрограммы в основную программу передаются результаты расчетов.

Использование процедур и функций позволяет: сократить объем программы, улучшить структуру программы с точки зрения ее читаемости, уменьшить вероятность ошибок и облегчает процесс создания и отладки программы.

В языке Турбо Паскаль подпрограммы реализуются в виде процедур и функций

**Процедуры**

Для использования процедуры в программе необходимы:

* описание процедуры;
* вызов процедуры.

Описание процедуры содержит информацию для компилятора об объектах, используемых в процедуре, а также алгоритм решения задачи процедурой.

Вызов процедуры приводит к выполнению операторов, составляющих тело процедуры.

***Описание процедуры***

Процедура в Турбо Паскале имеет структуру, подобную структуре программы. Структура описания процедуры имеет следующий вид:

**Procedure** <*имя процедуры*> (параметры); заголовок процедуры;

**Label** объявление меток;

**Const** объявление констант;

**Type** объявление типов;

**Var**  объявление переменных;

**procedure** описание процедур и функций, входящих в данную процедуру (внутренние процедуры и функции);

**function**

**Begin**

**Тело процедуры** операторы процедуры;

**End**; точка с запятой «;» ставится всегда

Таким образом, процедура состоит из:

***заголовка;***

***раздела описаний;***

***раздела операторов.***

Заголовок процедуры в отличие от заголовка программы не может быть опущен.

**Формат заголовка**

***Procedure*** <*имя*> (<*список формальных параметров*>);

где *Procedure* – зарезервированное слово;

*имя* – имя процедуры, определяемое в соответствии с общими правилами построения идентификаторов;

 <*список формальных параметров*> – перечень идентификаторов для обозначения исходных данных и результатов работы процедуры с указанием их типа.

Параметры в списке заголовка процедуры отделяются друг от друга точкой с запятой «;». Если параметры являются однотипными, их можно объединить и записать через запятую. Например: ***Procedure*** SUMMA(a,b:real; i,n:integer);

Здесь SUMMA имя процедуры, a,b,i,n формальные параметры процедуры.

Раздел описаний процедуры содержит объявление меток, констант, типов переменных и внутренних процедур и функций, используемых в разделе операторов процедуры.

Раздел операторов ограничен операторными скобками BEGIN и END. При описании процедуры после END всегда ставится точка с запятой «;».

**Вызов процедуры**

Если процедура объявлена, то в программе ее можно использовать многократно, просто задавая ее имя и, если необходимо, список аргументов, т.е. вызов происходит с помощью оператора вызова:

<*имя процедуры*> (<*список фактических параметров*>);

где <*имя процедуры*> – имя процедуры, к которой происходит обращение;

 <*список фактических параметров*> – перечень конкретных значений (выражений) и имен, подставляемых на место формальных параметров процедуры при ее выполнении.

При описании подпрограмм и вызова их используются понятия формальных и фактических параметров.

Параметры, указываемые в заголовке процедуры при ее описании, называются формальными параметрами. Параметры, указываемые при вызове процедуры, называются фактическими параметрами.

Формальные параметры – это параметры, определяющие тип и место подстановки реальных объектов при вызове процедуры.

При вызове процедуры формальные параметры, указанные в ее заголовке, заменяются на фактические параметры в порядке их следования. Число и тип формальных и фактических параметров должен обязательно совпадать.

***ЗАДАЧА 1. Ввести с клавиатуры 15 вещественных чисел в массив и найти сумму этих чисел с помощью процедуры.***

**Примечание: всё, что в разобранных для примера программах написано в фигурных скобках – это есть пояснение. При написании программ в практической работе следующего занятия это можно не писать.**

program a21;

Const n =15;

Type

mas=array[1..n] of real;

Var a:Mas;

i:integer;

{-----------------описание процедуры ----------------}

procedure summa(b:mas;t:integer);

{b-массив, t-количество чисел – формальные параметры, которые будут заменены на фактические параметры при вызове процедуры summa}

var

{--------объявление внутренних переменных------------}

i:integer; {параметр цикла}

s:real; {сумма чисел}

begin

s:=0;

for i:=1 to t do s:=s+b[i];

writeln;

writeln(‘Сумма равна:’, s:8:2);

end; {конец процедуры}

{----------------------программа---------------------}

begin

writeln(‘Введите ’, n:4,’ чисел’);

for i:=1 to n do

begin

write(‘a[‘,i,’]=’);

readln(a[i]);

end;

summa(a,n); {вызов процедуры summa, сумма n чисел}

summa(a,10); {вызов процедуры summa, сумма 10 чисел}

end.

***ЗАДАЧА 2. Найти наибольший элемент массива и определить его номер. Количество элементов массива и их значения ввести с клавиатуры. Оформить ввод данных, поиск наибольшего элемент массива и определение его номера с помощью процедур.***

program a22;

Const n =100; {допустимое количество чисел}

Type

mas=array[1..n] of real;

Var a:mas;

m:integer; {количество вводимых чисел}

{--------------процедура ввода массива---------------}

Procedure vvod\_mas(b:mas; var t:integer);

var

i:integer;

begin

write(‘Введите количество чисел: ’);

readln(t);

writeln(‘Введите ’, t:4,’ чисел’);

for i:=1 to t do

begin

write(‘b[‘,i,’]=’);

readln(b[i]); {ввод элементов массива}

end;

end; {конец процедуры}

{------процедура поиска максимального элемента-------}

Procedure max\_el\_array(b:mas; t:integer);

{b - массив, t- количество чисел}

var

i:integer; {параметр цикла}

m:real; {максимальный элемент массива}

k:integer; {его порядковый номер}

begin

m:=b[1]; k:=1; {присваиваем начальное значение}

for i:=2 to t do

if b[i]>m then

begin

m:=b[i]; k:=i;

end;

writeln;

writeln(‘Номер наибольшего элемента:’,k:3);

writeln(‘Его значение:’,m:8:2);

end; {процедуры}

{---------------------программа----------------------}

begin

vvod\_mas(a,m);

max\_el\_array(a,m);

end.

**Функции**

Подпрограмма-функция предназначена для вычисления одного параметра. У функции два основных отличия от процедуры

***Первое отличие*** в заголовке. Он состоит из слова Function , за которым следует имя функции, затем в круглых скобках список формальных параметров, затем через двоеточие записывается тип функции, т.е. тип возвращаемого параметра. Функция может возвращать типы вещественные, порядковые и строковые.

***Второе отличие*** в том, что процедура может иметь несколько выходных параметров-результатов, а функция только одно значение, передаваемое через ее имя. Именно этим объясняется то, что в теле функции хотя бы один раз имени функции должно присваиваться вычисленное значение.

Структура функции такая же, как и структура процедуры.

Заголовок функции имеет вид:

 ***Function*** <*имя*> (<*список формальных параметров*>): <*тип*>

где

*Function* – служебное слово;

<*имя*> – имя функции, определяемое в соответствии с общими правилами построения идентификаторов;

<*список формальных параметров*> – перечень имен для обозначения исходных данных и результатов работы процедуры с указанием их типов.

Для вызова функции из основной программы или другой подпрограммы следует в выражении, где необходимо использовать значение функции, указать имя функции со списком фактических параметров, которые должны совпадать по количеству и типам с формальными параметрами функции.

***ЗАДАЧА 3. Ввести с клавиатуры 15 вещественных чисел в массив и найти сумму этих чисел с помощью функции.***

program a23;

Const n=15;

Type

mas=array[1..n] of real;

Var

a:mas;

i:integer;

sum:real;

{-----------------описание функции-------------------}

function summa(b:mas; t:integer): real;

{b - массив}

{t - количество чисел}

{b, t - формальные параметры, которые будут заменены на фактические параметры при вызове функции summa}

var

{объявление внутренней переменной i – параметр цикла}

{объявление внутренней переменной s – суммы чисел}

i:integer;

s:real;

begin

s:=0;

for i:=1 to t do s:=s+b[i];

writeln;

{--------имени функции присваиваем результат!------}

summa:=s;

end; {конец функции}

{-----------------основная программа----------------}

begin

writeln(‘Введите ’, n:4,’ чисел’);

for i:=1 to n do

begin

write(‘a[‘,i,’]=’);

readln(a[i]);

end;

{вызов функции summa, для вычисления суммы n чисел}

sum:=summa(a,n);

writeln(‘Сумма равна:’, sum:8:2);

{вызов функции summa, для вычисления суммы 10 чисел}

sum:=summa(a,10);

writeln(‘Сумма равна:’, sum:8:2);

end.

**Примечание: До окончания изучения дисциплины «Основы алгоритмизации и программирования» остается всего три занятия, после чего вы будете сдавать экзамен. До сдачи экзамена все практические работы должны быть вами выполнены. Свои долги по работам вы знаете. Те студенты, кому были отправлены письма по поводу прочтения работ (не открывались ваши письма), должны переслать еще раз эти практические работы для проверки.**