МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ТУЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт прикладной математики и компьютерных наук

Кафедра информационной безопасности

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к лабораторным работам (часть 1)

по дисциплине

ПРОГРАММИРОВАНИЕ

Уровень профессионального образования: высшее образование – бакалавриат

Направление подготовки: *090303 Прикладная информатика*

Профиль подготовки: *Прикладная информатика в экономике*

Квалификация выпускника: *бакалавр*

Тула 2018 г.

ЛАБОРАТОРНАЯ РАБОТА №1

ИЗУЧЕНИЕ МЕНЮ ИНТЕГРИРОВАННОЙ СИСТЕМЫ.
НАЧАЛА ПРОГРАММИРОВАНИЯ (ПАСКАЛЬ

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Изучение меню и настройка интегрированной среды. Получение навыков по составлению и отладке простейших программ на языке Паскаль. Изучение организации простейшего ввода-вывода на языке Паскаль.

2. ОБЩИЕ ПОЛОЖЕНИЯ

Алфавит языка Паскаль.

При записи программ используются следующие символы:
- буквы латинского и русского алфавитов и арабские цифры;
- знаки препинания: <. , : ; ' >;
- знаки арифметических операций: + - = * /;
- знаки отношений: = <> > >= <= <; и скобки: () [];
- специальные знаки: ^ $ _;

Константы и переменные.

Константы - это величины, которые остаются неизменными в процессе выполнения программы. В Паскале существует два вида констант: именованные и литеральные. Именованная константа представляется в начале программы по имени и получает фиксированное заданное значение. Присваивание имен константам делает программу более удобной для понимания и внесения изменений. Литеральными константами являются числа, символы и строки символов. Числа могут быть целого и вещественного типов. Максимальная точность представления вещественных чисел - семь десятичных цифр. Символьные константы представляют собой любой символ, взятый в апострофы. Текстовые константы (строки символов) - это последовательность символов, взятая в апострофы.

Например:
Целые константы: 2, -567, 32000
Вещественные константы: 0.456, 56.89765, 34E-23, 567E13
Символьные константы: ' ', '7', 'G', '%'
Текстовые константы: 'FISHER', 'GH TYUI', '5+6-67'

Переменные - это величины, значения которых изменяются в процессе выполнения программы и обозначения которых осуществляется с помощью имен (идентификаторов). Имя всегда начинается с буквы, а если оно состоит из нескольких слов, то они разделяются только символами подчеркивания («_»). Для всех данных должны быть определены типы и они должны быть описаны в разделе Var.

Целый тип (INTEGER). Над данными целого типа могут производиться любые арифметические действия.

Вещественный тип (REAL). Вещественные числа характеризуются диапазоном и точностью (т.е. количеством знаков после запятой)

Логический тип (BOOLEAN). Переменные логического типа могут принимать только два значения - "истина" (TRUE) и "ложь" (FALSE) и использоваться в операциях сравнения.

Символьный (CHAR) и строковый (STRING) типы. Символьный тип – это тип данных, состоящих из одного символа (знака, буквы, кода). Запись символьного значения представляет собой сам символ, заключенный в кавычки.

Структура Паскаль-программы.

Программа на языке Паскаль состоит из следующих частей (рис.1.1)

Основной особенностью Паскаля является то, что все данные, с которыми работает программа, должны быть предварительно описаны в соответствующих разделах. Исполнительная часть программы начинается со слова BEGIN и заканчивается END. (с точкой). Операторы программы разделяются символом “ ; ”.

В состав программы могут включаться комментарии, которые заключаются в специальные ограничители (фигурные скобки {}) и могут размещаться в любом месте программы.

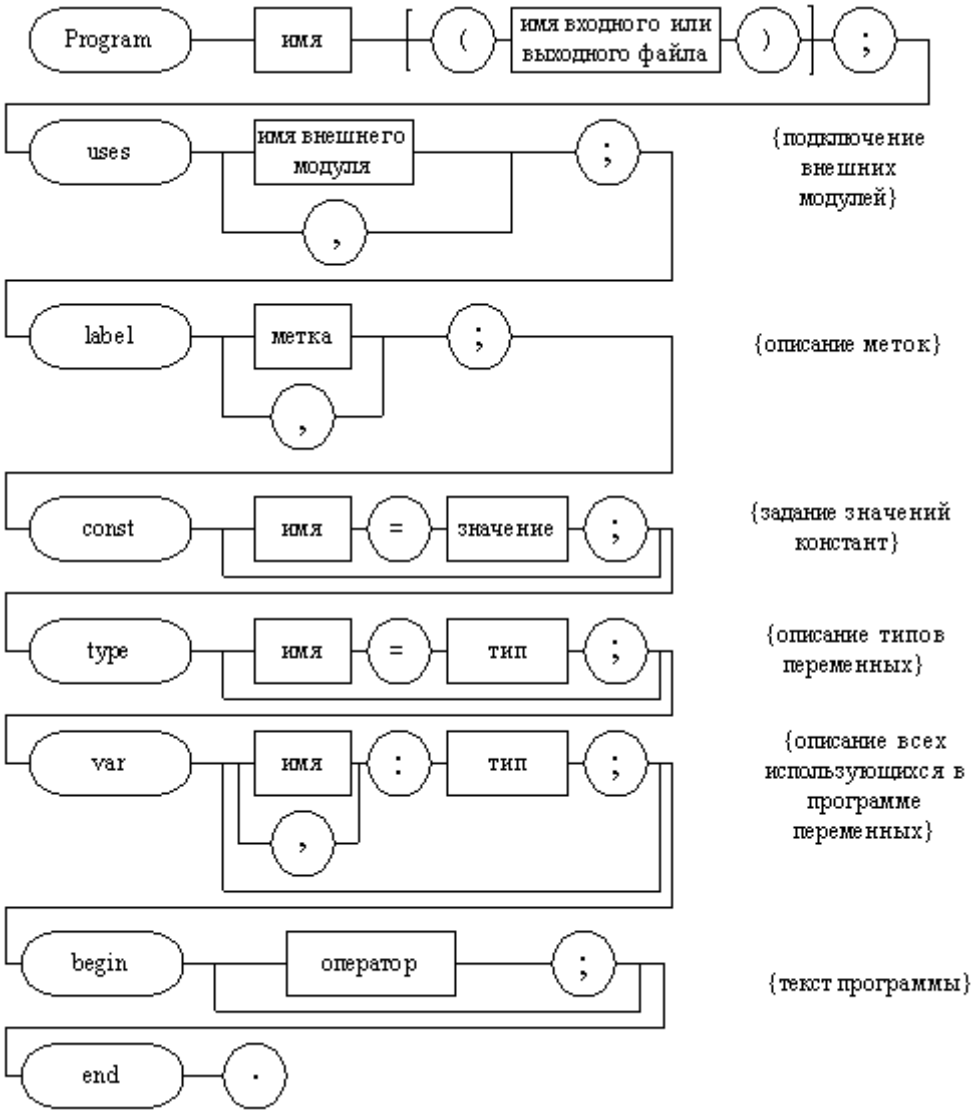


Рис.1.1.

Арифметические выражения.

Арифметические выражения состояний констант, переменных и функций, соединенных знаками арифметических операций. Порядок вычисления выражения определяется следующим приоритетом:

-выражения в скобках;

-стандартные функции;

-операции умножения, деления;

-операции сложения и вычитания.

Операции с одинаковыми приоритетами выполняются слева направо. Для выполнения часто встречающихся математических функций используются стандартные функции модуля MATH (табл.1.1).

Таблица 1.1.

Стандартные функции модуля MATH

Математические функции	Стандартные функции языка Паскаль	Математические функции	Стандартные функции языка Паскаль
$\|x\|$	ABS(X)	ln x	LN(X)
x^2	SQR(X)	cos x	COS(X)
$\sqrt{x}$	SQRT(X)	sin x	SIN(X)
e^x	EXP(X)	arctg x	ARTAN(X)
a^b	power(a,b)		

Составной оператор и оператор присваивания.

Группа операторов, ограниченная служебными словами BEGIN и END, называется составным оператором (рис.1.2).

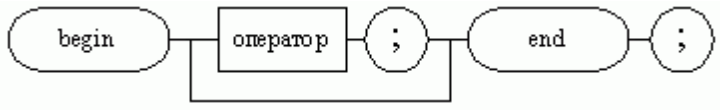


Рис.1.2.

Оператор присваивания предназначен для вычисления арифметических выражений и присваивания вычисленных значений переменным (рис.1.3).

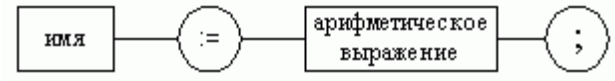


Рис.1.3.

Сначала вычисляется значение выражения из правой части оператора, а затем это значение присваивается переменной, имя которой записано в левой части оператора. Тип выражения должен быть совместим с типом переменной оператора присваивания.

Пример.

Вычислить функцию $z = \frac{\pi}{2}\sqrt{x^2 + 2{,}5} + \frac{\cos x^2}{\sin x - 0{,}01x}$, при x=0,8

```
PROGRAM PROBA;
CONST PI=3.1416;
      EPS=1E-2;
VAR Z, X: REAL;
BEGIN
```

```
X:=0.8;
Z:=PI/2.0*SQRT(SQR(X)+2.5);
Z:=Z+COS(SQR(X))/(SIN(X)-X*EPS);
WRITE('Z=',Z);
END.
```

Операторы ввода и вывода.

Операторы ввода и вывода данных обеспечивают программу исходными данными и осуществляют вывод результатов решения. Поэтому любая программа должна содержать эти операторы, выполненные в языке Паскаль как процедуры.

Ввод данных осуществляется следующими операторами (рис.1.4, рис.1.5)

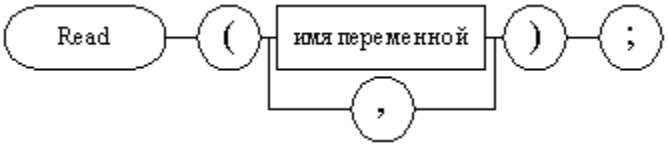


Рис.1.4.

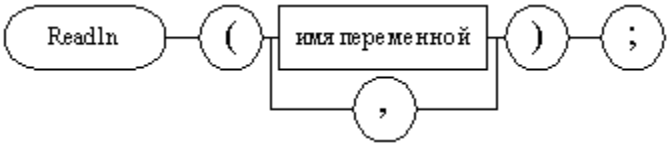


Рис.1.5.

При вводе обеспечивается выборка данных из входного файла, в результате чего имена переменных, указанные в списке ввода, получают соответствующие значения. Типы переменных должны соответствовать типам вводимых значений. Вводимые значения могут принадлежать к целому, вещественному или символьному типам. Ввод переменных логического типа не допускается. Вводимые числа отделяются друг от друга одним или несколькими пробелами в строке ввода или могут располагаться в разных строках. При вводе данных типа CHAR все символы, включая пробелы и переходы к новой строке, являются значимыми. Оператор READLN отличается от оператора READ тем, что после ввода значения последней переменной списка осуществляется переход к следующей строке ввода, т.е. оставшиеся в строке значения игнорируются. Оператор READLN без списка ввода осуществляет пропуск одной строки. Этот оператор "ловит" переход к следующей строке при вводе данных символьного типа.

Пример ввода данных для этой программы

```
PROGRAM
WWOD;
VAR A, B : REAL;
K, M : INTEGER;
BEGIN
READ(A,B);          12.3 3.25E+2
READLN(K);          25
READLN(M);          9
...
END.
```

Переход к новой строке осуществляется нажатием клавиши <Enter>. После ввода переменным будут присвоены следующие значения: A=12.3, B=325, K=25, M=9. Значения для переменных A и B можно вводить в разных строках, а ввод в одной строке значений для переменных K и M будет являться ошибочным, т.к. после ввода значения переменной K произойдет автоматический переход к новой строке.

После этого программа будет находиться в ожидании ввода значения переменной M, ранее введенное значение игнорируется.

Вывод данных осуществляется следующими операторами (рис.1.6, рис.1.7).

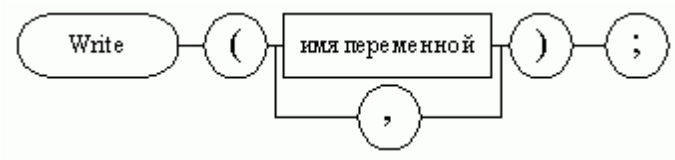


Рис.1.6.

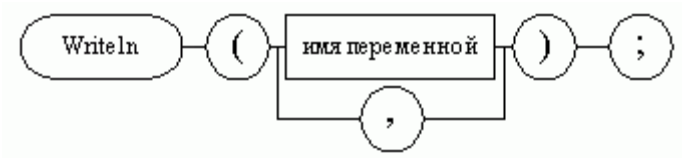
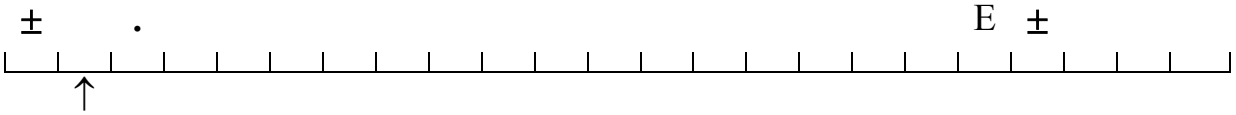


Рис.1.7.

Имена переменных, указанные в списке вывода, могут быть любого стандартного типа. Несколько операторов WRITE осуществляют вывод данных в одну строку. Оператор WRITELN после вывода последнего элемента списка осуществляет переход к следующей строке. Оператор WRITELN без списка обеспечивает переход к новой строке. Форма представления выводимых переменных и ширина поля вывода определяются типом переменных. Вещественные числа выводятся в форме с порядком (или в экспоненциальной форме) (рис.1.8). При этом задается одна цифра перед точкой, а под мантиссу отводятся остальные позиции. При выводе символьных данных каждый символ занимает одну позицию.



Любая значащая цифра, но не нуль

Рис.1.8.

При этом E±00N означает 10 в степени +N или -N.

Например, значения переменных A и K из предыдущего примера будут выведены с помощью оператора WRITE(A,K) в следующем формате:

3.250000E+02 25

В операторах вывода допускается использовать специальные форматы вывода, которые в явном виде задают ширину поля и определяют форму представления выводимых данных. Формат вывода может иметь вид:

:w

или

:w:d,

где w, d - константы или выражения целого типа. w задает общее количество позиций поля вывода, d определяет число цифр дробной части числа.

Для вывода значений целого типа используется только формат :w.

Вывод вещественных чисел можно производить по двум форматам. По формату :w:d вывод осуществляется в виде основной константы, а по формату :w - в виде константы с порядком.

Если при выводе число занимает меньше w позиций, то слева дополняются пробелы. Если же значение не помещается в заданную ширину поля, то для вывода будет отведено необходимое число позиций. В список вывода можно включать текстовую информацию, заключенную в апострофы, что позволяет формировать заголовки текстов, таблицы, выводить комментарии.

Многократное повторение пробела в строке вывода можно задать следующим образом, например, ' ':7, т.е. 7 пробелов. Если записать, например, '*':7, то в результате на экране будет 6 пробелов и символ "*" в седьмой позиции.

```
PROGRAM WYWOD;
 CONST R1=32.5; R2=0.92E+6;
         I1=137; I2=-23777;
         SIM='ТЕКСТ';
 BEGIN
 WRITELN(' *** КОНСТАНТЫ ***');
 WRITE(R1,R2); WRITE(I1,I2); WRITELN;
 WRITELN('R1=',R1:10:5);
 WRITELN('R2=',R2:10:3);
 WRITELN('I1=',I1:3,' ':5,'I2=',I2:5);
 WRITELN(SIM:7);
 END.
```

В результате выполнения программы на экран дисплея будет выдана информация в следующем виде:

```
*** КОНСТАНТЫ ***
3.250000000000000E+001 9.200000000000000E+005 137 -23777
R1= -32.50000
R2=920000.000
I1=137     I2=-23777
  ТЕКСТ
```

Интегрированная система

Интегрированная система состоит из трех основных областей (рис.1.9):

1. Главное меню
2. Окно редактирования
3. Строка состояния (подсказки)

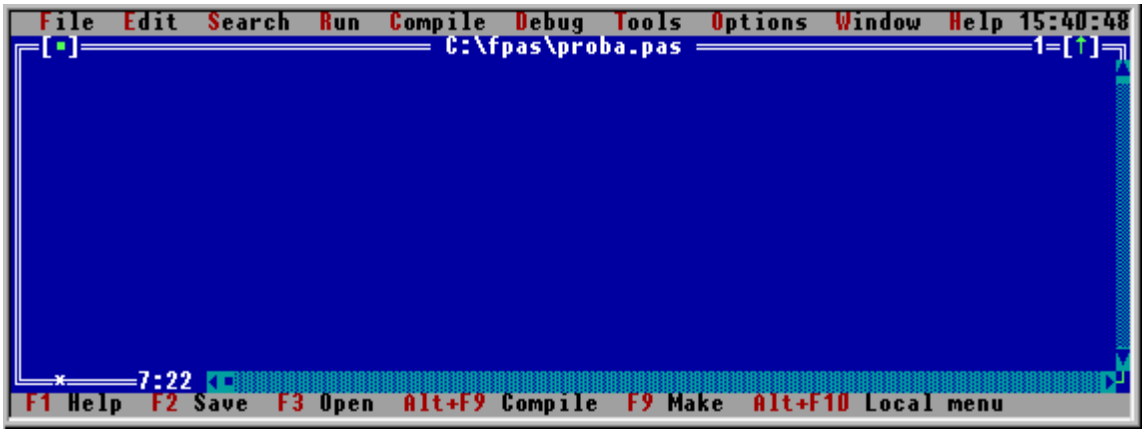


Рис.1.9.

Главное меню

Главное меню содержит основной набор действий для загрузки, редактирования, компиляции, компоновки (редактирования связей), отладки и выполнения программ.
Рассмотрим отдельные пункты и подпункты главного меню (табл.1.2).

Таблица 1.2.

Отдельные пункты и подпункты главного меню интегрированной системы

Пункты Главного Меню	Подпункты Главного Меню	Функции
File		**Работа с файлами**
	New	Создание нового файла
	Open F3	Загрузка файла
	Save F2	Сохранение файла, находящегося в редакторе, на диск, по умолчанию
	Save as...	Сохранение файла, находящегося в редакторе, под новым именем
	Change dir...	Отображает текущий каталог и позволяет назначить текущим другой дисковод и каталог
	Exit Alt+X[1]	Выход
Edit		**Редактор**
Search		**Поиск**
Run		**Выполнение**
	Run Ctrl+F9	Запуск программы на выполнение
	Step over F8	Пошаговое выполнение. При обращении к процедуре она будет выполняться как одна команда

[1] В дальнейшем знаком «+» будем обозначать одновременное нажатие клавиш

Пункты Главного Меню	Подпункты Главного Меню	Функции
	Trace into F7	Трассировать внутрь. При обращении к процедуре будет выполняться каждый оператор процедуры.
	Goto Cursor F4	Выполнять программу до курсора
Compile		**Компиляция (трансляция)**
	Compile Alt+F9	Компилировать в объектный файл
	Make F9	Создать выполняемый файл
	Target...	Устанавливает платформу (выходную операционную систему), для которой будет компилироваться
	Compiler messages F12	Сообщения компилятора
Debug		**Отладка**
	User screen Alt+F5	Просмотр результата работы программы
	Breakpoint Ctrl+F8	Установить точку прерывания в строке в месте расположения курсора.
	Add Watch Ctrl+F7	Ввод контролируемой переменной в окно Watches
	Watches	Отобразить список выражений окна Watches
Tools		**Инструментальные средства**
Options		**Опции**
	Directories	Пункты этого меню позволяют указать Free Pascal, где ему следует искать файлы для компиляции, компоновки, где размещать выходные файлы и где искать файлы конфигурации, помощи (help)
	Save fp.ini	Сохранить опции по умолчанию
	Save as...	Сохранить текущие опции
Window		**Работа с окнами**
	Zoom F5	Расширяет активное окно на весь экран. Повторное нажатие клавиши F5 переводит экран в режим наложения окон
	Next F6	Активизировать следующее окно в списке окон
	Previous Shift+F6	Активизировать предыдущее окно в списке окон
Help		**Помощь (справка)**
	Contents	Оглавление справки
	Index Shift+F1	Переход к справке индексов
	Topic search Ctrl+F1	Переход к теме, связанной с высвеченным текстом

Окно редактирования

В редакторе FreePascal осуществляется ввод и редактирование текста программы.

Чтобы войти в окно редактора необходимо выбрать пункт Edit главного меню.

В основной части окна редактора собственно редактируется файл; в центре в верхней части окна редактора указывается информация о редактируемом файле: имя файла с указанием пути (c:\fpas\proba.pas); в левом нижнем углу указываются номер строки и номер столбца (1:3) позиции, в которой находится курсор.

В процессе отладки программы появляется окно Watches

При редактировании используют команды пункта меню Edit или комбинацию следующих клавиш:

Ctrl+Y	Удаление строки
Ctrl+T	Удаление слова
Ctrl+K+B	Пометить начало блока
Ctrl+К+K	Пометить конец блока
Ctrl+К+V	Переместить блок
Ctrl+K+C	Скопировать блок
Ctrl+K+Y	Удалить блок

Составление и отладка программ

В общем случае для создания и отладки программ в среде Free Pascal необходимо выполнить следующие этапы:

- ввод и редактирование исходной программы;
- запись программы на диск;
- трансляция (компиляция) программы;
- выполнение программы.

Исходная программа - это совокупность следующих объектов: директив, указаний компилятору, объявлений и определений.

Исходная программа может содержать любое число директив, указаний компилятору, объявлений и определений. Любой из объектов программы имеет определенный синтаксис и каждая составляющая может появляться в любом порядке. Для записи программы выбирается соответствующая команда пункта меню File.

Для трансляции исходного файла необходимо выбрать в основном меню пункт Compile подпункт Compile или одновременно нажать клавиши Alt+F9.

Запуск программы осуществляется при выборе в основном меню пункта Run команды Run или одновременным нажатием клавиш Ctrl+F9.

Если в это время появились ошибки, необходимо их исправить и снова выполнить программу. Теперь компоновка программы должна завершиться успешно. Это значит, что для запуска программы копирует в нее необходимые для ее работы процедуры из библиотеки исполняющей системы.

В общем случае для создания и отладки программ в среде Lazarus необходимо выполнить следующие этапы:

3. ОБОРУДОВАНИЕ

ПЭВМ IBM PC, SVGA монитор с разрешением не менее 800*600 пикселей; клавиатура; мышь. Среда Free Pascal, Lazarus.

4. ЗАДАНИЯ НА РАБОТУ

4.1. Ознакомиться с теоретическими положениями лабораторной работы. Набрать в редакторе FreePascal или Lazarus приведенную ниже программу. Сохранить набранную программу на соответствующем диске. Выполнить необходимые настройки меню системы. Отладить, откомпилировать программу и выполнить программу.

```
{          Моя первая программа          }
PROGRAM nomer1;
Var x:Integer;
 y, z:Real;
BEGIN
WRITE (‘Введите значение переменной x: ’);
READLN(x);
WRITE (‘Введите значение переменной y:’);
READLN(y);
z:=y+x;
WRITELN (‘Сумма введенных переменных ’, z);
z:=x*y;
WRITE (‘Произведение введенных переменных ’, z:7:3);
END.
```

4.2. Разработать программу для решения следующей задачи. Даны катеты прямоугольного треугольника a и b. Найти гипотенузу c и периметр P

5. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретические положения.
2. Ввести программу с клавиатуры.
3. Отладить программу. Результаты работы программы показать преподавателю.
4. Оформить отчет.
5. Защитить лабораторную работу перед преподавателем.

6. СОДЕРЖАНИЕ ОТЧЕТА

1. Номер и название лабораторной работы
2. Цель и задачи
3. Текст программы
4. Результаты и выводы по лабораторной работе

7. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие символы входят в алфавит языка Паскаль?

2. Назовите основные разделы Паскаль-программы.

3. Что такое составной оператор?

4. Какой приоритет операций устанавливается при вычислении арифметических выражений?

5. Как выполняется оператор присваивания?

6. Что выделяется значками { }?

7. В чем отличие операторов READ и READLN?

8. В чем отличие операторов WRITE и WRITELN?

9. Поля какого размера отводятся по умолчанию при выводе переменных целого и вещественного типов?

10. В каком виде осуществляется вывод переменной вещественного типа по формату :w?

11. Каким образом можно задать многократное повторение пробела в строке вывода?

12. Как будет осуществляться вывод числа, если указанный размер поля недостаточен для размещения этого числа?

13. С помощью каких операторов осуществляется ввод-вывод данных? В чем особенность каждого оператора?

14. Как создать новый файл?

15. Как сохранить программу на внешнем носителе?

16. Как запустить программу на выполнение?

17. Как просмотреть результат выполнения программы?

ЛАБОРАТОРНАЯ РАБОТА №2

ОПЕРАТОРЫ ПЕРЕДАЧИ УПРАВЛЕНИЯ

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Приобретение навыков программирования алгоритмов разветвляющихся структур с использованием операторов передачи управления.

1. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Программы разветвляющейся структуры предусматривают выбор одной из нескольких последовательностей операторов в зависимости от некоторых условий.

Условный оператор

Условный оператор IF, используемый для реализации ветвлений в программе, может быть представлен в одной из двух форм (рис.2.1):

IF < Условие > THEN < Оператор 1 > ELSE < Оператор 2 >;

IF < Условие > THEN < Оператор 1 >;

где < Условие > - некоторое логическое выражение;

< Оператор 1 >, < Оператор 2 > - простые или составные операторы.

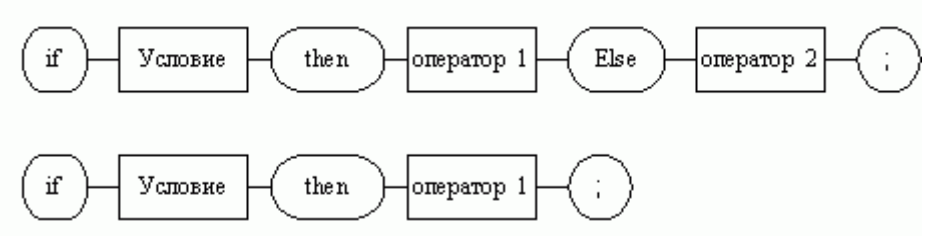


Рис.2.1.

Вторая форма оператора рассматривается как сокращение первой, где вместо альтернативного оператора стоит пустой оператор. Условие в операторе IF является логическим выражением, поэтому может включать логические переменные, выражения и арифметические отношения, соединенные знаками логических операций (=, <, >, <>, <=, >=, NOT, OR, AND).

В Паскале приоритет логических операций выше, чем приоритет операций сравнения. В условном операторе IF в качестве операторов могут в свою очередь использоваться операторы IF. В этом случае говорят о вложенной структуре? например:

```
…IF X<0
   THEN Y:=LN(X)
   ELSE IF X>5
          THEN Y:=EXP(X)
          ELSE Y:=SQR(X);…
```

Пример.

Если из отрезков с длинами X, Y, Z можно построить треугольник, то следует вычислить площадь этого треугольника по формуле s= p(p-x)(p-y)(p-z), где p=(x+y+z)/2 и X,Y, Z - положительные.

```
PROGRAM PRIM;
VAR X, Y, Z, S, P: REAL;
```

```
BEGIN
WRITELN('ВВЕДИТЕ ЗНАЧЕНИЯ X, Y, Z');
READ(X,Y,Z);
IF (X<0) OR (Y<0) OR (Z<0)
THEN WRITELN('СРЕДИ X, Y, Z ЕСТЬ ОТРИЦАТЕЛЬНЫЕ')
ELSE IF (X+Y>Z) AND (X+Z>Y) AND (Y+Z>X)
        THEN BEGIN
                P:=(X+Y+Z)/2.0;
                S:=SQRT(P*(P-X)*(P-Y)*(P-Z));
                WRITELN('ПЛОЩАДЬ ТРЕУГОЛЬНИКА =',S:8:2);
                END
        ELSE WRITELN('ТРЕУГОЛЬНИК ПОСТРОИТЬ НЕЛЬЗЯ');
WRITELN('ВЫПОЛНЕНИЕ ПРОГРАММЫ ЗАКОНЧЕНО')
END.
```

Оператор выбора

Оператор выбора CASE позволяет осуществить выбор одной из нескольких альтернатив, т.е. с помощью этого оператора осуществляется множественное ветвление. Оператор CASE состоит из выражения (селектора, ключа); констант, каждая из которых определен диапазоном, набором или одним значением (метка), и операторов. Тип значения константы должен совпадать с типом селектора (рис.2.2). Селектор может относиться к любому типу, кроме вещественного. Общая форма записи оператора выбора имеет вид:

```
CASE <селектор> OF
        <константа 1> : <оператор 1>;
        < константа 2> : <оператор 2>;
        ...
        < константа n> : <оператор n>;
        ELSE <оператор>;
END;
```

где < константа i> - список значений, разделенных запятыми (обязательно должно присутствовать хотя бы одно значение);

<оператор i> - простой или составной оператор.

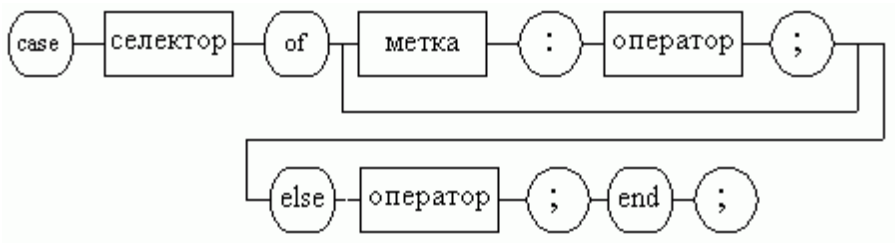


Рис.2.2

Значение селектора должно быть определено к моменту выполнения оператора. Оператор CASE осуществляет выбор того оператора, метка которого равна текущему значению селектора. По окончании выполнения выбранного оператора управление передается в конец оператора CASE. Если значение

селектора не соответствует ни одной из перечисленных меток варианта, то будет выполняться оператор, стоящий после ключевого слова ELSE.

Пример.

Произвести чтение числа от 1 до 7, преобразовать его в слово, соответствующее дню недели. При этом указать, является ли этот день выходным или рабочим.

```
PROGRAM NEDEL;
VAR DEN: INTEGER;
BEGIN
WRITE('ВВЕДИТЕ НОМЕР ДНЯ НЕДЕЛИ - ');
READ(DEN);
CASE DEN OF
        1: WRITE('ПОНЕДЕЛЬНИК');
        2: WRITE('ВТОРНИК');
        3: WRITE('СРЕДА');
        4: WRITE('ЧЕТВЕРГ');
        5: WRITE('ПЯТНИЦА');
        6: WRITE('СУББОТА');
        7: WRITE('ВОСКРЕСЕНЬЕ');
        ELSE WRITELN('НОМЕР ВНЕ ДИАПАЗОНА 1-7');
END;
CASE DEN OF
        1, 2, 3, 4, 5: WRITELN(' - РАБОЧИЙ ДЕНЬ');
        6, 7:          WRITELN(' - ВЫХОДНОЙ ДЕНЬ ');
        END;
END.
```

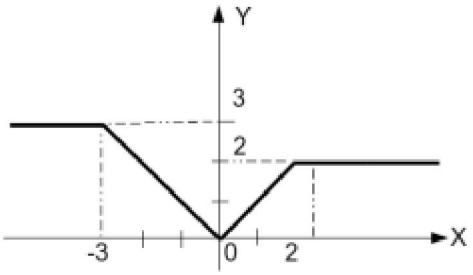
3. ОБОРУДОВАНИЕ

ПЭВМ IBM PC, SVGA монитор с разрешением не менее 800*600 пикселей; клавиатура; мышь. Среда Free Pascal, Lazarus.

4. ЗАДАНИЕ НА РАБОТУ

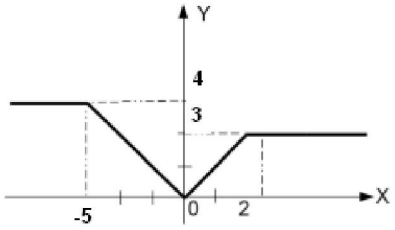
Номер варианта работы соответствует последней цифре номера зачетки

0. Составить программу, работающую в двух режимах: в первом режиме производится нахождение максимального из A, B, C и D, во втором - функции Y, заданной с помощью графика.



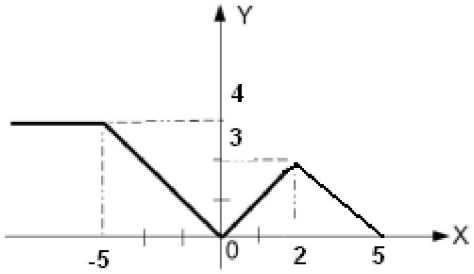
Выбор режима осуществить оператором CASE. Переменные A, B, C, D - целого типа.

1. Составить программу, работающую в двух режимах: в первом режиме производится нахождение максимального из A, B, C и D, во втором - функции Y, заданной с помощью графика.



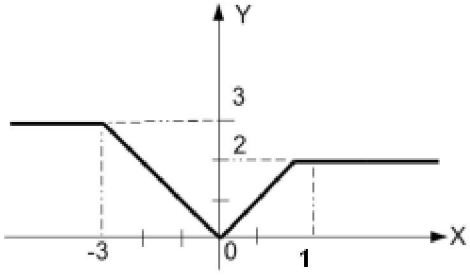
Выбор режима осуществить оператором CASE. Переменные A, B, C, D - целого типа.

2. Составить программу, работающую в двух режимах: в первом режиме производится нахождение максимального из A, B, C и D, во втором - функции Y, заданной с помощью графика.



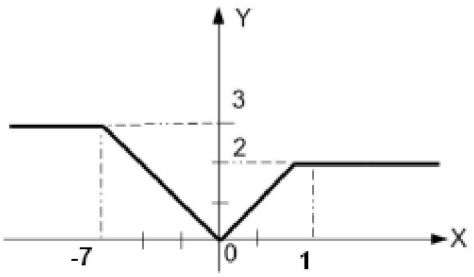
Выбор режима осуществить оператором CASE. Переменные A, B, C, D - целого типа.

3. Составить программу, работающую в двух режимах: в первом режиме производится нахождение максимального из A, B, C и D, во втором - функции Y, заданной с помощью графика.



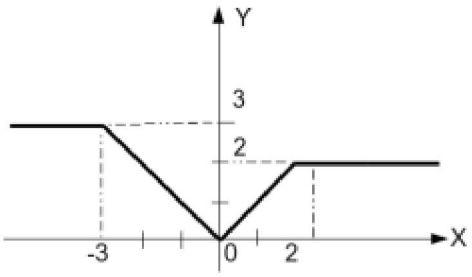
Выбор режима осуществить оператором CASE. Переменные A, B, C, D - целого типа.

4. Составить программу, работающую в двух режимах: в первом режиме производится нахождение максимального из A, B, C и D, во втором - функции Y, заданной с помощью графика.



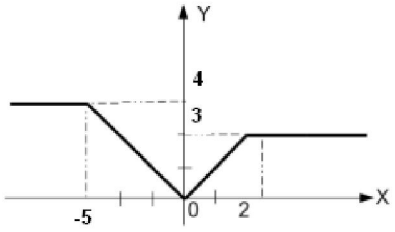
Выбор режима осуществить оператором CASE. Переменные A, B, C, D - целого типа.

5. Составить программу, работающую в двух режимах: в первом режиме производится нахождение минимального из A, B, C и D, во втором - функции Y, заданной с помощью графика.



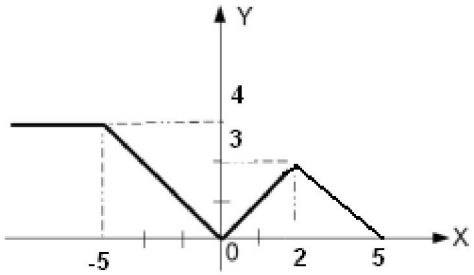
Выбор режима осуществить оператором CASE. Переменные A, B, C, D - целого типа.

6. Составить программу, работающую в двух режимах: в первом режиме производится нахождение минимального из A, B, C и D, во втором - функции Y, заданной с помощью графика.



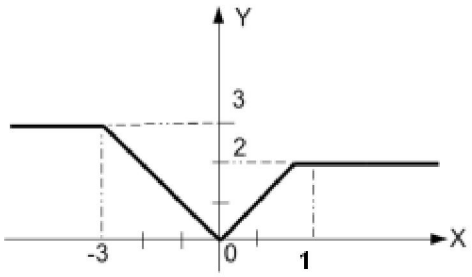
Выбор режима осуществить оператором CASE. Переменные A, B, C, D - целого типа.

7. Составить программу, работающую в двух режимах: в первом режиме производится нахождение минимального из A, B, C и D, во втором - функции Y, заданной с помощью графика.



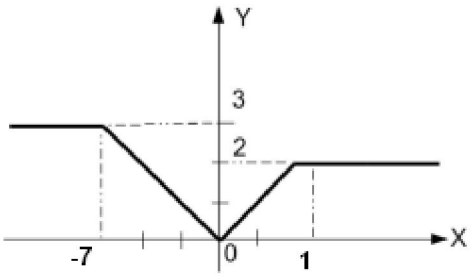
Выбор режима осуществить оператором CASE. Переменные A, B, C, D - целого типа.

8. Составить программу, работающую в двух режимах: в первом режиме производится нахождение минимального из A, B, C и D, во втором - функции Y, заданной с помощью графика.



Выбор режима осуществить оператором CASE. Переменные A, B, C, D - целого типа.

9. Составить программу, работающую в двух режимах: в первом режиме производится нахождение минимального из A, B, C и D, во втором - функции Y, заданной с помощью графика.



Выбор режима осуществить оператором CASE. Переменные A, B, C, D - целого типа.

5. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретические положения.
2. Разработать схему программы решения задачи.
3. Составить программу.
4. Отладить программу. Результаты работы программы показать преподавателю.
5. Оформить отчет.
6. Защитить лабораторную работу перед преподавателем.

6. СОДЕРЖАНИЕ ОТЧЕТА

1. Номер и название лабораторной работы.

2. Цель и задачи.
3. Задание на работу. Описание задания в соответствии с вариантом.
4. Схема программы.
5. Текст программы.
6. Результаты и выводы по лабораторной работе.

7. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Каков приоритет операций, используемых для записи условия в операторе IF?
2. Что такое вложенная структура IF?
3. Какой тип может иметь селектор и метки вариантов в операторе выбора?
4. Где описываются метки вариантов оператора CASE?
5. Каково назначение конструкции ELSE в операторе выбора?

ЛАБОРАТОРНАЯ РАБОТА №3

ОПЕРАТОРЫ ЦИКЛА.
РЕГУЛЯРНЫЙ ТИП ДАННЫХ (МАССИВЫ)

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Приобретение практических навыков программирования алгоритмов циклической структуры с использованием операторов цикла.

Приобретение практических навыков работы с регулярными типами данных (массивами).

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Операторы цикла

Программа циклической структуры позволяет организовать многократное выполнение группы операторов при изменении одного или нескольких параметров.

Оператор цикла с параметром.

Оператор цикла с параметром служит для организации цикла с известным числом повторений (рис.3.1). Общий вид записи оператора:

FOR <пц>:=<нз> TO <кз> DO <оператор>;

или

FOR <пц>:=<нз> DOWNTO <кз> DO <оператор>;

где пц - параметр цикла (переменная целого типа),

нз и кз - соответственно начальное и конечное значения параметра цикла (константа или переменная целого типа).

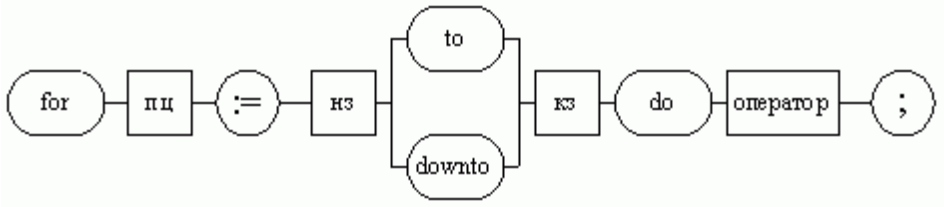


Рис.3.1.

Если в записи оператора используется ключевое слово TO, то значение параметра цикла увеличивается на единицу, а если DOWNTO - то уменьшается на единицу. На каждом шаге изменения параметра цикла выполняется оператор (простой или составной), расположенный после DO.

Пример.

Вычислить функцию $y = \sum_{i=1}^{n} i + \prod_{i=1}^{n} i$.

```
PROGRAM CIKL;
VAR I, N, SUM, PR: INTEGER;
BEGIN
SUM:=0;
PR:=1;
WRITE('ВВЕДИТЕ ЗНАЧЕНИЕ n - ');
READLN(N);
```

```
{   НАЧАЛО ЦИКЛА   }
FOR I:=1 TO N DO
                    BEGIN
                    SUM:=SUM+I;
                    PR:=PR*I;
                    END;
{   ОКОНЧАНИЕ ЦИКЛА   }
WRITELN('Y= ',SUM+PR:5);
END.
```

Оператор цикла с предусловием.

Оператор цикла с предусловием используется при организации цикла с неизвестным числом повторений (рис.3.2). Общий вид записи оператора:

WHILE <условие> DO <оператор>;

где <условие> - логическое выражение,

<оператор> - простой или составной оператор.

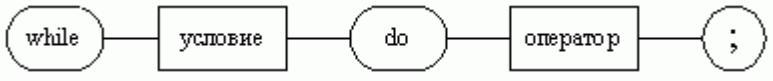


Рис.3.2.

Если условие выполняется (логическое выражение истинно), то повторяется оператор после слова DO. Выход из цикла осуществляется в случае невыполнения логического условия. Если при входе в оператор цикла условие ложно, то оператор после слова DO не выполняется ни разу, а управление передается следующему оператору программы.

Пример.

Вычислить значение функции y=x+3x+4 при изменении x от 0.2 до 1.8 с шагом 0.1.

```
PROGRAM TAB;
VAR X, Y: REAL;
      XN, XK, H: REAL;
BEGIN
WRITELN('ВВЕДИТЕ XN, XK, H');
READ(XN,XK,H);
WRITELN('ТАБУЛИРОВАНИЕ ФУНКЦИИ');
WRITELN;
WRITELN(' X Y');
(* ОРГАНИЗАЦИЯ ЦИКЛА *)
X:=XN;
WHILE X<=XK DO
BEGIN
      Y:=X*X+3.0*X+4.0;
      WRITELN(' ':3,X:3:1,' ':6,Y:5:2);
      X:=X+H;
END;
```

END.

Оператор цикла с постусловием.

Оператор цикла с постусловием также используется при организации цикла с неизвестным числом повторений (рис.3.3). Общий вид записи оператора:

REPEAT
 <оператор 1>;
 <оператор 2>;
 ...
 <оператор n>
 UNTIL <условие>;

где <условие> - логическое выражение,
 <оператор i> - простой оператор.

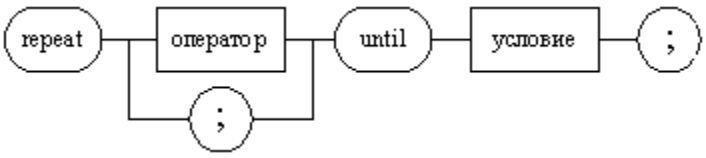


Рис.3.3

При выполнении условия осуществляется выход из цикла. Если условие не выполняется, то повторяются операторы цикла.

Пример. Ввести строку текста и определить число повторений буквы А. В качестве признака конца текста используется символ точка.

```
PROGRAM SIMVOL;
CONST A='A';
VAR SIM: CHAR;
        KA: INTEGER;
BEGIN
KA:=0;
WRITELN('ВВЕДИТЕ СТРОКУ ТЕКСТА');
REPEAT
        READ(SIM);
        IF SIM=A THEN KA:=KA+1
UNTIL SIM='.';
WRITELN;
WRITELN('БУКВА А ВСТРЕЧАЕТСЯ В ТЕКСТЕ ',KA:2,' РАЗ');
END.
```

В общем случае внутри одного цикла можно организовать один или несколько других циклов (внутренних). Такие циклы называются вложенными.

Регулярный тип данных (массивы)

Регулярные типы данных представляют собой упорядоченную последовательность компонент одного типа, называемого базовым. Каждая компонента характеризуется индексом, который определяет местоположение компоненты в массиве. Регулярный тип данных может быть описан в разделе определения типов следующим образом:

TYPE T = ARRAY [T1] OF T2;

где T - имя типа;

T1 - тип индексов (обычно интервальный от целого типа);

T2 - тип компонент (базовый), который может быть любым типом Паскаля.

Имена массивов задаются в разделе описания переменных.

Пример. Объявить вещественный массив A, состоящий из 100 элементов.

```
TYPE MAS=ARRAY[1..100] OF REAL;
VAR A : MAS;
```

Массив может быть описан непосредственно в разделе VAR без использования вспомогательного имени типа в разделе TYPE.

Пример.

```
VAR A : ARRAY[1..100] OF REAL;
```

При обращении к массиву в программе нужно указать имя массива и индекс элемента в квадратных скобках. Индекс может задаваться константой, переменной или выражением того же типа, что и тип индекса, например, A[12], A[N], A[I+2].

Если массивы принадлежат к одному типу и содержат одинаковое количество компонент, то значения элементов одного массива могут быть присвоены элементам другого.

Например, если массивы описаны как

```
VAR A, B: ARRAY[1..10] OF INTEGER;
```

то в программе допустим следующий оператор присваивания: …A:=B;…

В этом случае компоненты массива B будут продублированы в массиве A. В языке Паскаль кроме одномерных массивов допускается использование многомерных массивов. При описании многомерных массивов в квадратных скобках задаются типы индексов, разделенные запятыми.

Например, описание матрицы целого типа MATR, состоящей из 5 строк и 10 столбцов, будет иметь вид:

```
VAR MATR : ARRAY[1..5,1..10] OF INTEGER;
```

Описание трехмерного массива C вещественного типа задается аналогичным образом:

```
VAR C : ARRAY[1..5,1..5,1..10] OF REAL;
```

При обращении к компонентам такого массива указывается имя массива и в квадратных скобках, через запятую, - значения соответствующих индексов, например, C[3,2,5]. Многомерные массивы запоминаются в памяти ЭВМ в виде одномерных, при этом каждая компонента массива занимает одну ячейку памяти. При работе с массивами целого и вещественного типа такой способ расположения является эффективным.

Пример. Организовать ввод элементов матрицы A(5,5) вещественного типа и определить среднее арифметическое ее элементов, а также количество элементов, превышающих это значение.

```
PROGRAM MASSIV;
```

```
CONST N=5;
VAR A: ARRAY[1..N,1..N] OF REAL;
        SR: REAL;
        I, J, KOL: INTEGER;
BEGIN
SR:=0;
{ ВВОД МАТРИЦЫ И ОПРЕДЕЛЕНИЕ SR }
FOR I:=1 TO N DO
BEGIN
        WRITELN('ВВЕСТИ ',I:1,' СТРОКУ');
        FOR J:=1 TO N DO
        BEGIN
        READ(A[I,J]);
        SR:=SR+A[I,J]
        END
END;
SR:=SR/(N*N);
{ ПОДСЧЕТ ЭЛЕМЕНТОВ, БОЛЬШИХ SR }
FOR I:=1 TO N DO
        FOR J:=1 TO N DO
        IF A[I,J]>SR THEN KOL:=KOL+1;
{ ВЫВОД РЕЗУЛЬТАТОВ }
WRITELN;
WRITELN(' SR=',SR:6:1);
WRITELN(KOL:2,' ЭЛЕМЕНТОВ > SR')
END.
```

3. ОБОРУДОВАНИЕ

ПЭВМ IBM PC, SVGA монитор с разрешением не менее 800*600 пикселей; клавиатура; мышь. Среда Free Pascal, Lazarus.

4. ЗАДАНИЯ НА РАБОТУ

Задание 1. **Номер варианта работы соответствует последней цифре номера зачетки**

0. Написать 3 варианта программы с использованием операторов FOR, WHILE, REPEAT UNTIL для решения следующей задачи

Задано целое положительное число n. Определить значение выражения:

$$P = \frac{\sum_{i=3}^{n} i - 2}{(n+1)} + \sum_{i=2}^{n-1} (i^3).$$

1. Написать 3 варианта программы с использованием операторов FOR, WHILE, REPEAT UNTIL для решения следующей задачи

Задано целое положительное число n. Определить значение выражения:

$$P = \frac{\sum_{i=3}^{n} i - 2}{(n+1)} + n^3 .$$

2. Написать 3 варианта программы с использованием операторов FOR, WHILE, REPEAT UNTIL для решения следующей задачи

Задано целое положительное число n. Определить значение выражения:

$$P = \frac{\sum_{i=3}^{n} i^i - 2}{(n+1)} .$$

3. Написать 3 варианта программы с использованием операторов FOR, WHILE, REPEAT UNTIL для решения следующей задачи

Задано целое положительное число n. Определить значение выражения:

$$P = \frac{\sum_{i=3}^{n} i - \sqrt{\mathrm{i}} * 2}{(n+1)} .$$

4. Написать 3 варианта программы с использованием операторов FOR, WHILE, REPEAT UNTIL для решения следующей задачи

Задано целое положительное число n. Определить значение выражения:

$$P = \frac{\sum_{i=3}^{2+n} i - 2}{(n+1)!2} .$$

5. Написать 3 варианта программы с использованием операторов FOR, WHILE, REPEAT UNTIL для решения следующей задачи

Задано целое положительное число n. Определить значение выражения:

$$P = \frac{\sum_{i=3}^{n} i - 2}{(n+1)!n} - \sum_{i=0}^{n} i .$$

6. Написать 3 варианта программы с использованием операторов FOR, WHILE, REPEAT UNTIL для решения следующей задачи

Задано целое положительное число n. Определить значение выражения:

$$P = \frac{\sum_{i=3}^{n} i - 2^{\mathrm{n}}}{(n+1)!n} .$$

7. Написать 3 варианта программы с использованием операторов FOR, WHILE, REPEAT UNTIL для решения следующей задачи

Задано целое положительное число n. Определить значение выражения: $P = \frac{\sum_{i=3}^{n} i - 2}{(n+1)! \sum_{i=1}^{n} n}$.

8. Написать 3 варианта программы с использованием операторов FOR, WHILE, REPEAT UNTIL для решения следующей задачи

Задано целое положительное число n. Определить значение выражения: $P = \frac{\sum_{i=3}^{n} i - 2}{(n+1)! \sum_{i=0}^{n} i^2}$.

9. Написать 3 варианта программы с использованием операторов FOR, WHILE, REPEAT UNTIL для решения следующей задачи

Задано целое положительное число n. Определить значение выражения: $P = \frac{\sum_{i=3}^{n} i - 2}{(n+1)!(n-5)}$.

4.2. Задание 2. **Номер варианта работы соответствует последней цифре номера зачетки**

0. Разработать программу на языке Паскаль для решения следующей задачи.
Задан массив целых чисел X(n). Найти
• сумму чётных элементов массива;
• наибольшее из отрицательных чисел массива.
Из данного массива и некоторого массива того же типа, но другой размерности Y (m), сформировать общий массив Z(n + m).

1. Разработать программу на языке Паскаль для решения следующей задачи.
Задан массив целых чисел X(n). Найти
• сумму нечётных элементов массива;
• наибольшее из отрицательных чисел массива.
Из данного массива и некоторого массива того же типа, но другой размерности Y (m), сформировать общий массив Z(n + m).

2. Разработать программу на языке Паскаль для решения следующей задачи.
Задан массив целых чисел X(n). Найти
• сумму чётных элементов массива;
• наименьшее из отрицательных чисел массива.
Из данного массива и некоторого массива того же типа, но другой размерности Y (m), сформировать общий массив Z(n + m).

3. Разработать программу на языке Паскаль для решения следующей задачи.

Задан массив целых чисел X(n). Найти

• сумму нечётных элементов массива;

• наименьшее из отрицательных чисел массива.

Из данного массива и некоторого массива того же типа, но другой размерности Y (m), сформировать общий массив Z(n + m).

4. Разработать программу на языке Паскаль для решения следующей задачи.

Задан массив целых чисел X(n). Найти

• сумму чётных элементов массива;

• наибольшее из положительное чисел массива.

Из данного массива и некоторого массива того же типа, но другой размерности Y (m), сформировать общий массив Z(n + m).

5. Разработать программу на языке Паскаль для решения следующей задачи.

Задан массив целых чисел X(n). Найти

• сумму нечётных элементов массива;

• наибольшее из положительное чисел массива.

Из данного массива и некоторого массива того же типа, но другой размерности Y (m), сформировать общий массив Z(n + m).

6. Разработать программу на языке Паскаль для решения следующей задачи.

Задан массив целых чисел X(n). Найти

• сумму чётных элементов массива;

• наименьшее из положительное чисел массива.

Из данного массива и некоторого массива того же типа, но другой размерности Y (m), сформировать общий массив Z(n + m).

7. Разработать программу на языке Паскаль для решения следующей задачи.

Задан массив целых чисел X(n). Найти

• сумму нечётных элементов массива;

• наименьшее из положительное чисел массива.

Из данного массива и некоторого массива того же типа, но другой размерности Y (m), сформировать общий массив Z(n + m).

8. Разработать программу на языке Паскаль для решения следующей задачи.

Задан массив целых чисел X(n). Найти

• сумму чётных и нечетных элементов массива;

• последнее из положительное чисел массива.

Из данного массива и некоторого массива того же типа, но другой размерности Y (m), сформировать общий массив Z(n + m).

9. Разработать программу на языке Паскаль для решения следующей задачи.

Задан массив целых чисел X(n). Найти

• сумму нечётных и четных элементов массива;

• первое из положительное чисел массива.

Из данного массива и некоторого массива того же типа, но другой размерности Y (m), сформировать общий массив Z(n + m).

5. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретические положения.
2. Разработать схему программы решения задачи.
3. Составить программу.
4. Отладить программу. Результаты работы программы показать преподавателю.
5. Оформить отчет.
6. Защитить лабораторную работу перед преподавателем.

6. СОДЕРЖАНИЕ ОТЧЕТА

1. Номер и название лабораторной работы
2. Цель и задачи
3. Задание на работу. Описание задания в соответствии с вариантом.
4. Схема программы
5. Текст программы
6. Результаты и выводы по лабораторной работе

7. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В каких случаях используется оператор цикла FOR?
2. К какому типу данных должен относиться параметр цикла в операторе FOR?
3. Когда в операторе FOR используется ключевое слово DOWNTO?
4. Какие операторы используются для организации циклов с неизвестным числом повторений?
5. Укажите различия между операторами WHILE и REPEAT?
6. В каких случаях удобнее пользоваться оператором WHILE?
7. В каких случаях удобнее пользоваться оператором REPEAT?
8. К какому типу данных может принадлежать тип индексов при описании массивов?
9. Каким образом может быть описан массив?
10. Можно ли описать массив в разделе VAR?
11. Как хранятся многомерные массивы в памяти ЭВМ?

ЛАБОРАТОРНАЯ РАБОТА №4

ПРОЦЕДУРЫ И ФУНКЦИИ

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Приобретение практических навыков организации подпрограмм на языке Паскаль.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

При программировании часто возникает необходимость выделять неоднократно повторяемые действия в виде отдельных частей программы - подпрограмм. Кроме того, использование подпрограмм позволяет реализовать принцип модульного построения программы, что значительно облегчает ее проектирование и отладку. Подпрограммы реализуются в языке Паскаль в виде процедур и функций. Структура подпрограмм эквивалентна структуре программ - состоит из заголовка и блока, имеющего описательную и исполнительную части.

При работе с подпрограммами возникает понятие локальных и глобальных параметров. Все параметры, определенные внутри подпрограммы, являются локальными. Вне области подпрограммы они не действуют и являются неопределенными. Параметры, описанные в начале программы, называются глобальными и доступны в любом месте программы, в том числе и в подпрограммах. При организации подпрограмм рекомендуется выбирать разные имена локальных и глобальных параметров, так как это делает программу нагляднее. Если используется одинаковое имя для глобального и локального параметра, то глобальный параметр теряет свое значение в подпрограмме, так как с этим именем будет связан локальный параметр.

Процедура

Процедура является подпрограммой общего вида, т.е. используется в тех случаях, когда необходимо получить несколько выходных результатов. Определение процедур осуществляется после раздела VAR в разделе процедур и функций. Описание процедуры начинается с ключевого слова PROCEDURE, за которым следует имя процедуры и список формальных параметров, заключенных в круглые скобки. Список параметров может и отсутствовать. В списке перечисляются имена формальных параметров и определяются их типы. Список параметров может содержать параметры четырех видов: параметры-значения, параметры-переменные, параметры-процедуры и параметры-функции.

Параметры-значения определяют исходные данные для процедуры, которые нежелательно изменять в процессе выполнения процедуры.

Параметры-переменные, как правило, определяют выходные данные процедуры, поэтому в списке параметров им должно предшествовать слово VAR.

Параметрами-процедурами и параметрами-функциями называются параметры, которые сами являются процедурами и функциями.

Пример заголовка процедуры:

```
PROCEDURE PRIM (I,J: INTEGER; L:REAL; VAR S,K: REAL);
```

где I,J,L - параметры-значения,
S,K - параметры переменные.

Если в процедуру передается массив, то при описании в списке процедуры он должен относиться к типу, имеющему имя. Например, будет ошибочным заголовок процедуры:

```
PROCEDURE PREI (VAR MAS: ARRAY[1..100] OF REAL);
```

Правильной формой записи будет следующая:

```
TYPE VEST= ARRAY[1..100] OF REAL;

PROCEDURE PREI (VAR MAS: VEST);
```

т.е. в случае описания массива требуется предварительное объявление типа. После заголовка процедуры следуют описательная часть, в которой определяются все локальные параметры, и исполняемая часть процедуры. Начинается процедура словом BEGIN и заканчивается словом END с точкой с запятой (END;).

Вызов процедуры осуществляется в программе с помощью оператора процедуры, который содержит имя процедуры и список фактических параметров, заключенных в круглые скобки. Фактические параметры должны согласовываться с формальными параметрами по типу, количеству и порядку следования. Процедура может быть описана в теле программы, а также может быть оформлена в виде внешнего модуля.

Пример. Составить процедуру определения суммы и произведения элементов строки матрицы. Используя эту процедуру найти сумму и произведение элементов второй строки матрицы A(5,5) вещественного типа.

```
PROGRAM PROC;
CONST N=5;
VAR A : ARRAY[1..N,1..N] OF REAL;
      SUM, PR : REAL;
      I, J : INTEGER;
{-------------------------------------}
{  НАЧАЛО ПРОЦЕДУРЫ  }
PROCEDURE MATR(NS: INTEGER; VAR S, P: REAL);
{  NS – номер строки  }
{  S – сумма элементов строки  }
{  P – произведение элементов строки  }
VAR K : INTEGER;
BEGIN
S:=0;
P:=1;
FOR K:=1 TO N DO
    BEGIN
```

```
      S:=S+A[NS,K];
      P:=P*A[NS,K];
      END;
END;
{   КОНЕЦ ПРОЦЕДУРЫ   }
{-----------------------------------}
BEGIN
(* ВВОД ЭЛЕМЕНТОВ МАТРИЦЫ A *)
FOR I:=1 TO N DO
BEGIN
WRITELN('ВВЕДИТЕ ',I:1,' СТРОКУ');
FOR J:=1 TO N DO
READ(A[I,J]);
END;
{   ОПРЕДЕЛЕНИЕ СУММЫ (SUM) И ПРОИЗВЕДЕНИЯ (PR) 2 - ОЙ СТРОКИ   }
MATR(2,SUM,PR);
WRITELN('SUM=',SUM:5:1,' PR=',PR:7:1);
END.
```

Как видно из примера, кроме локальных параметров, в процедуре используются глобальные параметры - константа N и массив A. Однако, для организации циклов в процедуре допускается использование только локальных параметров, т.е. в операторе FOR процедуры переменная цикла должна быть обязательно определена как локальная.

Функции

Функция является частным случаем процедуры и обычно используется для описания подпрограмм, в результате выполнения которых вычисляется одно значение, присваиваемое имени функции. Описание функции начинается с ключевого слова FUNCTION, за которым следует имя функции и список формальных параметров, заключенный в круглые скобки. Заканчивается заголовок описанием типа функции, который следует за списком параметров. Относительно списка параметров функции имеется полная аналогия с параметрами процедуры. Примеры записи заголовка функций:

```
FUNCTION REM( M: REAL; SEC: INTEGER) : INTEGER;
FUNCTION SIM( A: INTEGER) : CHAR;
```

Необходимо помнить, что имени функции обязательно должен быть присвоен результат, полученный в процессе выполнения функции. Вызов функции осуществляется указателем, который записывается в выражении. Указатель задается именем функции, за которым в скобках перечисляются фактические параметры. Оформление внешней функции аналогично оформлению внешней процедуры.

Пример. Составить программу для определения числа сочетаний $C = \frac{n!}{m!(n-m)!}$, используя функцию для вычисления факториала:

```
PROGRAM FUN;
VAR N, M: INTEGER;
        CMN: REAL;
FUNCTION FACT(K: INTEGER): INTEGER;
VAR P, I: INTEGER;
BEGIN
P:=1;
FOR I:=1 TO K DO
         P:=P*I;
FACT:=P
END;
BEGIN
WRITELN('ВВЕДИТЕ N И M');
READ(N,M);
CMN:=FACT(N)/(FACT(M)*FACT(N-M));
WRITELN;
WRITELN('ЧИСЛО СОЧЕТАНИЙ CMN= ',CMN)
END.
```

3. ОБОРУДОВАНИЕ

ПЭВМ IBM PC, SVGA монитор с разрешением не менее 800*600 пикселей; клавиатура; мышь. Среда Free Pascal, Lazarus.

4. ЗАДАНИЕ НА РАБОТУ

Номер варианта работы соответствует последней цифре номера зачетки

Составить программу с использованием подпрограмм и функций.

0. В двумерном массиве A[10, 7] найти минимальный элемент каждой из строк и максимальный элемент каждого из столбцов. Для поиска минимального использовать подпрограмму, которая находит минимальной элемент в одной строке, а для поиска максимального - функцию которая находит максимальный элемент в одном столбце.

1. В двумерном массиве A[7, 7] найти максимальный элемент каждой из строк и максимальный элемент каждого из столбцов. Для поиска минимального использовать подпрограмму, которая находит минимальной элемент в одной строке, а для поиска максимального - функцию которая находит максимальный элемент в одном столбце.

2. В двумерном массиве A[5, 8] найти минимальный элемент каждой из строк и минимальный элемент каждого из столбцов. Для поиска минимального

использовать подпрограмму, которая находит минимальной элемент в одной строке, а для поиска максимального - функцию которая находит максимальный элемент в одном столбце.

3. В двумерном массиве A[15, 8] найти минимальный и максимальный элемент каждой из строк и минимальный элемент каждого из столбцов. Для поиска минимального использовать подпрограмму, которая находит минимальной элемент в одной строке, а для поиска максимального - функцию которая находит максимальный элемент в одном столбце.

4. В двумерном массиве A[5, 10] найти минимальный элемент каждой из строк и количество положительных элементов каждого из столбцов. Для поиска минимального использовать подпрограмму, которая находит минимальной элемент в одной строке, а для поиска максимального - функцию которая находит максимальный элемент в одном столбце.

5. В двумерном массиве A[10, 10] найти минимальный элемент каждой из строк и количество отрицательных элементов каждого из столбцов. Для поиска минимального использовать подпрограмму, которая находит минимальной элемент в одной строке, а для поиска максимального - функцию которая находит максимальный элемент в одном столбце.

6. В двумерном массиве A[8, 8] найти количество положительных элементов каждой из строк и максимальный элемент каждого из столбцов. Для поиска минимального использовать подпрограмму, которая находит минимальной элемент в одной строке, а для поиска максимального - функцию которая находит максимальный элемент в одном столбце.

7. В двумерном массиве A[10, 8] найти количество отрицательных элементов каждой из строк и максимальный элемент каждого из столбцов. Для поиска минимального использовать подпрограмму, которая находит минимальной элемент в одной строке, а для поиска максимального - функцию которая находит максимальный элемент в одном столбце.

8. В двумерном массиве A[11, 11] найти количество отрицательных элементов каждой из строк и количество положительных элементов каждого из столбцов. Для поиска минимального использовать подпрограмму, которая находит минимальной элемент в одной строке, а для поиска максимального - функцию которая находит максимальный элемент в одном столбце.

9. В двумерном массиве A[11, 11] найти количество положительных элементов каждой из строк и количество отрицательных элементов каждого из столбцов. Для поиска минимального использовать подпрограмму, которая находит минимальной элемент в одной строке, а для поиска максимального - функцию которая находит максимальный элемент в одном столбце.

5. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретические положения.
2. Разработать схему программы решения задачи.
3. Составить программу.
4. Отладить программу. Результаты работы программы показать преподавателю.

5. Оформить отчет.
6. Защитить лабораторную работу перед преподавателем.

6. СОДЕРЖАНИЕ ОТЧЕТА

1. Номер и название лабораторной работы
2. Цель и задачи
3. Задание на работу. Описание задания в соответствии с вариантом.
4. Схема программы
5. Текст программы
6. Результаты и выводы по лабораторной работе

7. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Чем отличаются локальные и глобальные параметры?
2. Почему не рекомендуется выбирать одинаковые имена локальных и глобальных параметров?
3. Какие виды параметров может содержать список параметров?
4. Как осуществляется определение массива в списке параметров?
5. Каким образом оформляется внешняя процедура?