МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ

«БРАТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Кафедра электроэнергетики и электротехники

УТВЕРЖДАЮ:
Проректор по учебной работе
_____Е.И.Луковникова
«_____» декабря 2018 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

Б1.В.07

НАПРАВЛЕНИЕ ПОДГОТОВКИ

13.03.02 Электроэнергетика и электротехника

ПРОФИЛЬ ПОДГОТОВКИ
Электроснабжение

Программа академического бакалавриата

Квалификация (степень) выпускника: бакалавр

СОДЕРЖАНИЕ ПРОГРАММЫ **Стр.**

1. ПЕРЕЧЕНЬ ПЛАНИРУЕМЫХ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ, СООТНЕСЕННЫХ С ПЛАНИРУЕМЫМИ РЕЗУЛЬТАТАМИ ОСВОЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Вид деятельности выпускника

Дисциплина охватывает круг вопросов, относящихся к производственно-технологическому виду профессиональной деятельности выпускника в соответствии с компетенциями и видами деятельности, указанными в учебном плане.

Цель дисциплины

Формирование знаний о принципах организации и практической реализации компьютерных технологий в электроэнергетических системах.

Задачи дисциплины

Усвоение студентами основных принципов создания, разработки, отладки и тестирования алгоритмов и программ.

Код компетенции	Содержание компетенций	Перечень планируемых результатов обучения по дисциплине
1	2	3
ОПК-1	Способность осуществлять поиск, хранение, обработку и анализ информации из различных источников и баз данных, представлять ее в требуемом формате с использованием информационных, компьютерных и сетевых технологий	**знать:** - способы использования компьютерных технологий в своей предметной области. **уметь:** - применять компьютерную технику в своей профессиональной деятельности, в работе над проектами электроэнергетических и электротехнических систем и отдельных их компонентов. **владеть:** - средствами компьютерной техники и сетевых технологий в своей предметной области .
ПК-2	Способность обрабатывать результаты эксперимента	**знать:** – методы обработки экспериментальных данных. **уметь:** - проводить анализ экспериментальных данных. **владеть:** – навыками обработки экспериментальных данных.

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Дисциплина Б1.В.07 Компьютерные технологии относится к базовой.

Дисциплина Компьютерные технологии базируется на знаниях, полученных при изучении таких учебных дисциплин, как: Информатика, Физика.

Основываясь на изучении перечисленных дисциплин, Компьютерные технологии представляет основу для изучения дисциплин: Приемники и потребители электрической энергии систем электроснабжения, Основы проектирования систем электроснабжения.

Такое системное междисциплинарное изучение направлено на достижение требуемого ФГОС уровня подготовки по квалификации бакалавр.

3. РАСПРЕДЕЛЕНИЕ ОБЪЕМА ДИСЦИПЛИНЫ

3.1. Распределение объема дисциплины по формам обучения

Форма обучения	*Курс*	*Семестр*	*Трудоемкость дисциплины в часах*						*Курсовая работа*	*Вид промежуточной аттестации*
			Всего часов (с экз.)	*Аудиторных часов*	*Лекции*	*Лабораторные работы*	*Практические занятия*	*Самостоятельная работа*		
1	2	3	4	5	6	7	8	9	10	11
Заочная	1	-	108	10	4	-	6	94	КРз	зачет

3.2. Распределение объема дисциплины по видам учебных занятий и трудоемкости

Вид учебных занятий	*Трудоемкость (час.)*	*в т.ч. в интерактивной, активной, инновационной формах, (час.)*	*Распределение по семестрам, час*
			1
I. Контактная работа обучающихся с преподавателем (всего)	10	2	10
Лекции (Лк)	4	2	4
Практические занятия (ПЗ)	6	-	6
Контрольная работа	+	-	+
II.Самостоятельная работа обучающихся (СР)	94	-	94
Подготовка к практическим занятиям	66	-	66
Подготовка к зачету	18	-	18
Выполнение контрольной работы	10		10
III. Промежуточная аттестация зачет	+	-	+
Общая трудоемкость дисциплины, час.	108	2	108
зач. ед.	3		3

4. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

4.1. Распределение разделов дисциплины по видам учебных занятий

- для заочной формы обучения:

№ раздела и темы	*Наименование раздела и тема дисциплины*	*Трудоемкость, (час.)*	*Виды учебных занятий, включая самостоятельную работу обучающихся и трудоемкость; (час.)*		
			учебные занятия		*самостоятельная работа обучающихся*
			лекции	*практические занятия*	
1	2	3	4	5	6
1.	**Общие сведения и элементы языка**	**3**	**1**	**-**	**2**
1.1.	Общие сведения	1	0,5	-	0,5
1.2	Элементы языка	2	0,5	-	1,5
2.	**Структура программы**	**10**	**1**	**2**	**7**
2.1.	Правила записи программы	3,5	0,5	1	2
2.2.	Процедуры. Порядок операторов	6,5	0,5	1	5
3.	**Система в/в и встроенные функции**	**9**	**1**	**2**	**6**
3.1.	Система ввода/вывола	5,5	0,5	1	4
3.2.	Встроенные функции	3,5	0,5	1	2
4	**Операторы и графические функции**	**82**	**1**	**2**	**79**
4.1.	Операторы	51,5	0,5	1	50
4.2.	Графические функции	30,5	0,5	1	29
	ИТОГО	**104**	**4**	**6**	**94**

4.2. Содержание дисциплины, структурированное по разделам и темам

Раздел 1. Общие сведения и элементы языка

Тема 1.1. Общие сведения

Лекция проводится в интерактивной форме: лекция-беседа (0,5 час).

Алгоритмический язык FORTRAN создан в 50-е годы в США фирмой IBM , группой разработчиков под руководством Джона Бэкуса (коммерческая версия выпущена в 1957 г.).

FORmula + TRANslator = FORTRAN

(переводчик формул на машинный язык)

Разработан специально для решения инженерных и научно-технических задач. Целью разработки было создание языка, обладающего удобством программирования, характерным для языков высокого уровня и быстродействием, характерным для языков низкого уровня. В результате получилось, что программы созданные с помощью оптимизирующих компиляторов FORTRANa по эффективности практически не уступают программам, написанным с помощью машинных кодов или ассемблеров.

Существуют 5 стандарта языка:

1. FORTRAN 66 – принят в 1966 г.;
2. FORTRAN 77 – принят в 1978 г.;
3. FORTRAN 90 – принят в 1991 г.;
4. FORTRAN 95 – принят в 1997 г.;
5. FORTRAN 2003 – принят в 2005 г.

Готовится к принятию новый стандарт. Его основные отличия к поддержке объективно-ориентированного программирования и совместного использования с языком СИ.

Помимо стандартов существуют различные версии языка. Версии – это адаптация стандарта под конкретную вычислительную систему:

1. базисный FORTRAN – «Искра 1256»;
2. FORTRAN IV – СМ ЭВМ ;
3. FORTRAN –Дубна – БЭСМ – 6;
4. FORTRAN ЕС ЭВМ – ЕС ЭВМ.

В России наиболее распространены версии стандарта FORTRAN 77, разработанные различными формами для основных операционных систем:

1. Microsoft FORTRAN V5.0 и V5.1 для DOS;
2. Microsoft FORTRAN Powerstation - для Windows;
3. Visual FORTRAN Pro - для Windows фирмы COMPAQ;
4. INTEL FORTRAN compiler V4.5 для DOS;
5. WATCOM FORTRAN compiler V9.5 для DOS.

FORTRAN выпускают и другие фирмы, в том числе:
Heqlet Packard;
IBM; Sun Microsystems.

Существуют версии языка для операционных систем семейства UNIX (в частности Linux); Power Mac, а так же для суперкомпьютеров Cray, Sparc.

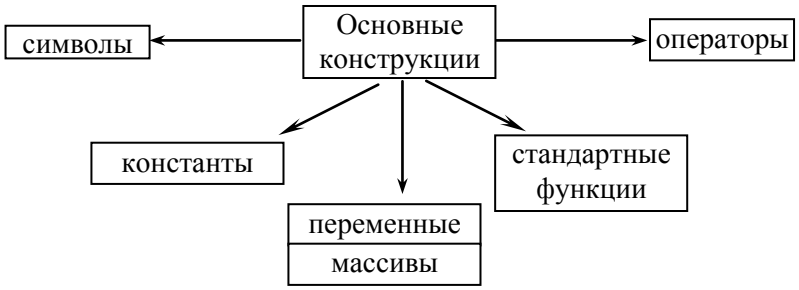
Наиболее важным преимуществом FORTRANa является наличие большого числа библиотек математических и графических подпрограмм, например: NAG (Numerical Algorithms Group).

Некоторые версии языка и библиотеки распространяются свободно, их можно найти в Интернете.

Тема 1.2. Элементы языка

Лекция проводится в интерактивной форме: лекция-беседа (1,5 час).

Основные конструкции языка FORTRAN



Символы языка (алфавит)

1. Буквы латинского алфавита A ÷ Z.

Особо выделим: I, J, K, L, M, N - смысл станет ясным, когда будем рассматривать типы переменных.

2. Цифры- 0 ÷ 9.
3. Специальные символы: + - * / () . , ′ ␣(знак «пробел») = и некоторые другие.
4. При записи комментарий можно использовать любые другие символы. Например, буквы русского алфавита.

Константы

- это величины задаваемые своим постоянным значением. Обычно это числа. Константы бывают различных типов.

- 100
- 100.1 ; 1.1E+2 ; 1.1E10
(1.3, + 4.5) = 1,3+j4,5
.TRUE. или .FALSE.
‘пример’ , 6Нпример
ZF

Шестнадцатеричные используются только для задания исходных значений.

Переменные

- это величина, которая может принимать различные значения. Задаётся с использованием имени.

Имя переменной

- состоит из различных символов, начинается обязательно с буквы и не должно превышать определённой длины. Длина зависит от версии FORTRANa. Рекомендуется употреблять ≤ 8.

X, A, B1, SUMMA.

Описание типов переменных

Переменные могут быть тех же типов, что и константы: целые, вещественные, комплексные, логические, текстовые.

С каждым типом переменной связано понятие стандартной и нестандартной длины, определяющее количество байт памяти, которые будет занимать переменная в памяти.

Тип переменной	Длина, байт	
	стандартная	нестандартная
целый	4	2
вещественный	4	8
комплексный	8	16
логический	4	1
удвоенной точности	8	-

Длина переменной имеет важное значение при обращении к подпрограммам. Нестандартная длина для целых (логических) переменных используется для экономии памяти; для вещественных и комплексных – для увеличения точности вычислений.

Для задания типа переменной существуют способы:

1. по умолчанию (по согласованию).

Выше были отмечены буквы I, J, K, L, M, N – если тип переменных начинающихся с этих букв не определён другим способом, то они считаются целыми, остальные - действительными.

Пример описания типа переменной по соглашению

```
I = 1        I = 1.1
a = 1.0      a = 1
 stop         stop
 end          end
```

2. с помощью операторов описания типа.

Операторы описания типа бывают двух видов: явного описания и неявного. Общая форма операторов описания типа (явного)

тип * l имя (группа имён)

где тип - INEGER, REAL, COMPLEX, LOGICAL, PHARACTER
(целый, вещественный, комплексный, логический, текстовый)

* l - длина переменной (для переменных стандартной длины можно опустить).
имя – имя переменной, либо массива.

Пример:

```
INEGER *2    A, B, C
REAL         I, D, E
REAL *8  A   →аналогично← DOUBLE PRECISION A.
```

В операторе описания типа можно задать начальные значения переменных, например:

```
NEGER *2    A/1/, B, C
```

Переменной A присвоено значение 1. Однако такой приём практически не используется, т.к. для этого существует специальный оператор DATA, кроме того, программа получается более наглядной.

Общая форма неявного описания типа:

IMPLICIT {тип длина список}

список – содержит один или несколько символов латинского алфавита (буквы)

оператор IMPLICIT позволяет описывать типы переменных по первому символу имени.

Например:
IMPLICIT INEGER (A, B) REAL (I – L)
все переменные, имена которых начинаются с букв A, B будут иметь тип целый; все переменные, имена которых начинаются от I до L (I, J, K, L) – будут иметь тип вещественный.
Примечание: рекомендуется использовать операторы явного описания типа.

Массивы

Массивы – это упорядоченная последовательность величин, обозначенных одним именем. (обычно занимает непрерывную область памяти).
Например:

A(1,1)	A (1,2)
A (2,1)	A (2,2)

a_{11}	a_{12}
a_{21}	a_{22}

массив записывается с помощью имени и круглых скобок, где указаны индексы. Любая переменная является массивом из одного элемента. Количество измерений массива не следует принимать более семи.

Индексы в скобках могут задаваться либо числом, либо переменной, либо выражением.
A (3, 7, к+5).

Имя массива образуется так же как имя переменной.

Объявление массивов осуществляется оператором
DIMENSION имя (размерность)

Размерность – определяет число элементов в каждом измерении массива.

DIMENSION A(3,4) B(5,10,15)

Типы массивов аналогичны типам переменных и задаются таким же образом:

1. по первой букве имени массива
2. с помощью операторов описания типа.

С помощью операторов описания типа можно определить и размерность массива:

INTEGER A(4), B(5,5)

REAL *8 C(10)

Это наиболее распространённый способ описания массивов.

Расположение массивов в памяти: элементы массива располагаются в памяти так, что значения первого индекса возрастает быстрее, чем значение последнего.
Пример: расположения в памяти двумерного массива

A(1,1)	A (1,2)
A (2,1)	A (2,2)

A(1,1)	A (2,1)	A (1,2)	A (2,2)

Существует жаргонное выражение: «массивы хранятся в памяти по столбцам». Поэтому, если выполнить оператор

PRINT A

то будет напечатано

A(1,1) A(2,1) A(1,2) A(2,2)

Раздел 2. Структура программы

Тема 2.1. Правила записи программы

Лекция проводится в интерактивной форме: лекция-беседа (2 час).

Правила записи одной строки:

Программа на FORTRANe состоит из последовательно записанных операторов. Один оператор может записать 20 строк с помощью специального знака переноса. Длина одной строки не более 80 символов (позиций). 80 - это число позиций на экране. С помощью редактора текста можно использовать и позиции с большими номерами, но они будут игнорироваться транслятором FORTRANa.

Строки длиной короче 72 символов "дополняются" пробелами.

В языке ФОРТРАН используются пять типов строк:

Строки комментария

Строка комментария имеет звездочку или прописную или строчную букву C в первой позиции, либо строка может полностью состоять из пробелов. Знак восклицания, появляющийся как первый непустой символ в строке, также указывает на строку комментария в языке ФОРТРАН. (Строка со знаком восклицания в колонке 6 интерпретируется как строка комментария).

Строка комментария на процесс выполнения ФОРТРАН-программы не влияет.

Строки комментария могут указываться внури операторов, которые имеют строки продолжения.

Следует отметить, что отладочные строки, описанные ниже, иногда могут быть обработаны как строки комментария.

Комментарии могут также добавляться к программным строкам. В таком случае комментарий должен начинаться со знака восклицания:

```
hyp = SQR (a**2 + b**2)  ! hypotenuse
```

Начальные строки

Первая (или единственная) строка оператора языка ФОРТРАН называется "начальной" строкой.

Начальная строка оператора в колонке 6 имеет либо пробел либо ноль; в колонках 1-5 указывается либо метка оператора либо пробелы.

За исключением оператора, следующего за логическим IF, все операторы ФОРТРАНа начинаются с начальной строки, например:

```
GOTO 100
0002  CHARACTER*10 name
100  0CONTINUE
1000STOP  ' '
```

Метакоманды

В первой позиции строки метакоманды присутствует знак доллара ($). Метакоманда управляет работой компилятора языка ФОРТРАН, например:

```
$DEBUG:'pdq'
$DECLARE
$LINESIZE:132
```

Строки продолжения

Строка продолжения представляет собой строку, имеющую пробелы в колонках 1-5 и символ (отличный от пробела или нуля) в колонке 6. Строка продолжения увеличивает размер "участка", в который записывается оператор. Оператор может быть расширен для включения такого количества строк продолжения, сколько позволит доступная для использования область памяти. Если установлена метакоманда $STRICT, то в том случае, если используется более 19 строк продолжения, будет выдано предупреждение. В приведенном ниже примере вторая строка является строкой продолжения:

```
C     Sample Continuation line
```

```
INTEGER*4 count, popu, local,
+         ovrflo, incrs, provnc
```

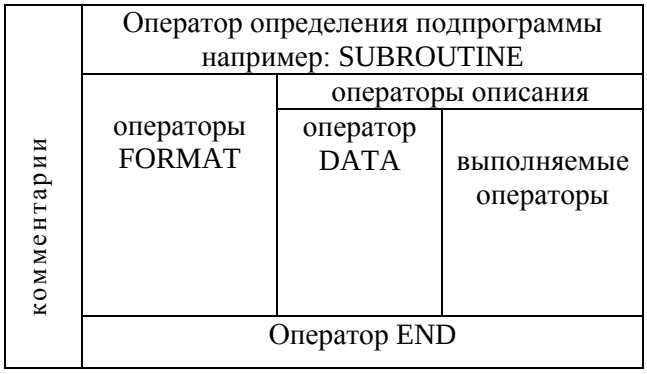
Отладочные строки Отладочные строки содержат любые прописные или строчные буквенные символы, за исключением C и c в позиции 1.
Если метакоманда $DEBUG специфицирует символ, находящийся в позиции 1 отладочной строки, то этот символ обрабатывается как символ пробела и строка компилируется как и любая другая строка. Если метакоманда $DEBUG не специфицирует символ в первой колонке, то строка обрабатывается как строка комментария (см. также подразд.6.2). В следующем примере строки являются строками отладки:

```
B   RETURN 1
Z   WRITE (*, *) count
```

Тема 2.2. Процедуры. Порядок операторов

Лекция проводится в интерактивной форме: лекция-беседа (6 час).

Порядок следования операторов в программе



При создании любой программы должен соблюдаться следующий порядок следования операторов:

1. оператор определения подпрограммы;
2. операторы описания. Среди них первым должен быть оператор IMPLICIT;
3. выполняемые операторы;
4. оператор END.

Операторы FORMAT и комментарии могут быть в любом месте программы. Однако для создания более удобной программы операторы FORMAT рекомендуется располагать в конце программы, перед оператором END. Оператор DATA может быть в любом месте среди выполняемых операторов. Рекомендуется располагать после операторов описания и до выполняемых операторов.

Раздел 3. Система в/в и встроенные функции

Тема 3.1. Система ввода/вывода

Операторы ввода - вывода

Операторы ввода-вывода предназначены для обмена информацией между носителями данных и основной памятью. Для ввода-вывода информации необходимо определить следующее:

1. с какого устройства (или на какое устройство) вводится –выводится информация.
2. значения каких переменных или массивов необходимо ввести – вывести.
3. в каком формате (виде) будет вводится - выводится информация.

Общая форма операторов ввода-вывода:

READ (n, m) s
WRITE (n, m) s
n- устройство ввода-вывода;
m-определитель формата ввода-вывода;
s- список ввода, вывода;
n- либо *, либо число (1 ÷99). Если это число-то считается номером файла на диске. Допуск этому файлу обеспечивается оператором OPEN:

```
OPEN (UNIT=, FILE= 'имя', STATUS='INKNOWN')
open (UNIT=1, FILE= 'TEST', STATUS='INKNOWN')
WRITE(1,*) A
```

Если n = *, то осуществляется ввод-вывод с экрана монитора:
FILE ='PRN' – принтер
FILE ='AUX' – порт COM1
FILE =' ' 'NUL' – считывается из командной строки.
Если m = *, то формат определяется по умолчанию. Если m= const, то это метка оператора FORMAT
WRITE (*,*) X - такая форма используется чаще всего для временной (отладочной) печати. Список ввода-вывода s чаще всего содержит переменные, массивы, текстовые сообщения.
WRITE (*,*) X,A, 'пример', (X(I))
Если используется оператор
READ (1,*)
без списка ввода-вывода, то пропускается одна запись файла с номером 1.

Пример.

```
R(Ом)      X(Ом)      B(МкСим)
10         1000       1500
      P(МВт)     Q(МВАР)
      3000       1000
```

вид файла №1

```
READ (1,*)
READ (1,*) K, X, B
READ (1,*)
READ (1,*) P,Q
```

текст на FORTRANe

Оператор FORMAT

Оператор FORMAT указывает, в каком виде должны вводится- выводится данные и для преобразования данных из внутреннего представления во внешнее и наоборот. Внутреннее представление – в виде двоичного числа.

n FORMAT (C_1, …….,C_n)

n-метка оператора FORMAT
C_1, …….,C_n – спецификация формата.
Термин «спецификация формата» - часто заменяют на просто «формат» имея ввиду не сам оператор, а спецификацию.

Форматы для преобразования данных разных типов

Тип переменной (массива)	Формат
Целый	Iw, Gwd
Вещественный (комплексный)	Fwd, Ewd, Dwd, Gwd, Qwd,
Логический	Lw, Gwd
Текстовый	Aw, wH, литерал
Шестнадцатеричный	Zw

w – количество позиций во внешнем представлении данных. Если число занимает меньше позиций, чем указано параметром w, то позиции слева заполняются пробелами;
d- количество цифр в дробной части числа или количество значащих цифр во внешнем представлении числа.

$7.3 \rightarrow 108.999$

- для преобразования данных вещественного типа из семи знаков (включая десятичную точку), из которых три отводятся под дробную часть. Число больше 1000. по такому формату передавать нельзя.

Если несколько соседних форматов имеют одинаковый вид, то их можно объединить, указав перед кодом формата количество повторений формата – r.

rIw

5F7.3 = F7.3, F7.3, F7.3, F7.3, F7.3.

Для повторения группы форматов она должна быть заключена в скобки:

3(I4, F5.1) = I4, F5.1, I4, F5.1, I4, F5.1

Передача данных целого типа

Iw, Gw.d

```
WRITE(*,1) N, K, L          N = 10
1 FORMAT (2I3, I5)          K = 1
                            L = - 300
   ␣10␣␣1␣-300
```

замечание: тип переменных ввода-вывода должен соответствовать коду формата.

Передача данных вещественного типа

Fw.d, Ew.d, Dw.d, Qw.d

Данные вещественного типа могут быть представлены в двух формах:

1. в виде десятичного типа, состоящего из целой и дробной части. Код формата F;
2. в виде мантиссы и порядка. Код формата E, D, Q.

12345.67 → F8.2

0,1234 E+02 →E10.4

␣ 0,1234 E+02 → E11.4

```
A = - 190,1      B = 10000,3
WRITE(*,1)  A, B
1 FORMAT(7.1,  7.1)
```

␣-190,1␣1,1E+04

Форматы E,D удобно использовать при передаче либо очень больших, либо близких к нулю данных. Для преобразования данных комплексного типа необходимо хорошо задавать два формата: один для действительной части, другой для мнимой.

```
COMPLEX     A
READ (*,1)  A
1 FORMAT(7.1,  7.1)
```

Передача данных логического типа

Lw

Логические переменные могут принимать значения либо .TRUE., либо .FALSE. Gосле ввода-вывода занимает w позиций и представляется в виде любой последовательности символов начинающихся с буквы T или F.

```
LOGICAL A                   LOGICAL A
READ (+, 1) A               A = .TRUE.
1 FORMAT(L3)                WRITE(+, 3) A
                            3 FORMAT(L1)
   ␣TR                          T
A = .TRUE.
```

Код формата G

Используется для передачи данных целого, вещественного и логического типов.

Передача данных целого и логического типов производится аналогично форматам I и L.

значение	тип	формат	экв. формат	Вывод (печать)
-125	INTEGER	G4.1	I4	-125
.TRUE.	LOGICAL	G3.2	L3	␣␣T

Вещественные данные по коду G вводятся так же, как по коду F, т.е. Gw.d = Fw.d

При вводе преобразование данных производится в зависимости от абсолютного значения данного n. Если значение данного n находится в диапазоне $0{,}1 < n < 10^d$, то форма вывода такая же, как по формату F. Если значение данного не принадлежит указанной области, то оно выводится по коду формат E или D. Форматы G удобно применять, когда основная часть данных имеет, «умеренные» значения, то есть также и очень маленькие, либо очень большие значения.

Передача текстовых данных

Aw

Если w не указано, то данное занимает столько позиций, какова длина соответствующего элемента ввода-вывода

```
CHARACTER *3 A
A = ‘ИМЯ’
WRITE(*,1) A
   1 FORMAT(A4)


 ␣ ИМЯ
```

wH – параметр w указывает длину текстовой константы, следующей за кодом формата H.

```
WRITE(*,1)

1 FORMAT(4H␣ИМЯ)
   ␣ ИМЯ
```

Литерал - это текстовая константа, заключённая в кавычки

```
‘␣ИМЯ’
WRITE(*,1)
1 FORMAT(‘␣ИМЯ’ )
```

В случае многократного вывода неизменной текстовой константы удобно пользоваться кодом формата Aw, при передаче данных текстовых констант, состоящих из одинаковых знаков – wH, при кратких и нечасто используемых константах – литералом.

5(1H*)

Организация пробелов при вводе-выводе

Формат wX – используется для пропуска данных при вводе или для записи пробелов при выводе.

В списке ввода-вывода отсутствуют элементы, соответствующие коду формата X.

```
WRITE (*,1)
1 FORMAT(3X,‘ТАБЛИЦА’,5X,‘РЕЗУЛЬТАТОВ’,5X, ‘РАСЧЁТА’)
```

Разделители форматов

Разделителями форматов являются / , : .

Запятой разделяются форматы, относящиеся к элементам одной записи. Знаком «/» разделяются форматы, относящиеся к элементам разных записей.

```
WRITE (*,1) A,B,C          WRITE (*,1) A,B,C
1 FORMAT(F5.1, F5.1, F5.1) 1 FORMAT(F5.1/F5.1/F5.1)
A B C                          A
                               B
                               C
```

Если n знаков «/» стоит в начале или конце форматов, пропускается n записей, если в середине группы форматов, то n –1 записей.

```
WRITE (*,1) A,B,C
1 FORMAT(F5.1//F5.1/ F5.1)
```

A

B

C

Замечание: спецификации форматов могут записываться непосредственно в операторах ввода-вывода.

```
WRITE (*,1) I,J,K
1 FORMAT('переменные', 3I4)

WRITE (*,'(3I4)') I,J,K
```

Если количество позиций w отводимых под число в операторе формат указано недостаточно, то при выводе будут напечатаны ****.

```
WRITE (*,'(3I)') 10000
***
```

Взаимодействие оператора FORMAT со списком ввода-вывода

При организации ввода-вывода могут иметь место следующие соотношения.

1. количество переменных в списке равно количеству спецификаций форматов;
2. количество спецификаций больше, чем количество переменных в списке. В этом случае оставшиеся спецификации игнорируются;
3. количество переменных в списке больше количества спецификаций. В этом случае анализ формата возобновляется с началом спецификации формата.

```
WRITE (*,'(3I4, F5.1)') A, B, C, D, E
I4, I4, I4, F5.1, I4
```

Тема 3.2. Встроенные функции

Стандартные функции FORTRANa

Это программы, хранящиеся в библиотеке FORTRANa, и вызываемые по её имени. После имени необходимо указать список параметров. Перечень стандартных функций, а так же число параметров и их тип даются в специальных таблицах. В качестве параметра может быть константа, переменная или выражение.

ABC (X) - вычисление модуля X, |X|

SIN (x) – sin x ASIN (x) – arc sin x

COS (x) – cos x ACOS(x) – arc cos x

TAN (A) – tg x ATAN (x) – arc tg x

SQT (x) = $\sqrt{x}$

Аргумент тригонометрических функций задаётся только в радианах.

Выражения

В FORTRANe существуют два вида выражений: арифметические и логические.

Арифметическое выражение – это любая последовательность констант, переменных или стандартных функций разъединённых знаками арифметических действий и круглыми скобками.

Порядок выполнения действий такой же, как в математике:

1. вычисление стандартных функций
2. x x
3. x /
4. + -

пример: $\frac{a+b}{c+d}$ (a+b) /(c+d)

$\sin^2 x + \cos^2 x$ sin(x)**2 + cos(x)**2

$\sqrt{\left|x_i + e^{x_i}\right|}$ SQRT(ABS(X(I) +EXP(X(I))))

Замечания:

1. В выражениях могут использоваться переменные различных типов. Результат всегда преобразуется к типу, имеющему наибольший приоритет. Например, если в выражении присутствуют целые и вещественные переменные, то результат будет вещественным.

2. Для того, чтобы программа работала более быстро, действия следует выполнять переменными одного типа.

3. Если выполняются действия над целыми числами то результат будет целым числом. Остаток просто отбрасывается.

$n = \frac{11}{3}$ n = 3

4. При возведении в отрицательную степень целого числа результат будет целым числом.

5 ** (-2) = $\frac{1}{25}$ = 0

Логические выражения

Рассмотрим только выражения отношения, т.к. остальные используются достаточно редко. Выражения отношения образуются из двух арифметических (или текстовых) выражений, разделённых операцией отношения. Выражения отношения используются для сравнения двух выражений.

Основные операции отношения

математическое обозначение	запись на FORTRANe
>	.GT.
≥	.GE.
<	.LT.
≤	.LE.
=	.EQ.
≠	.NE.

A .GT. B	a >b
(COS(X) +SIN(X)).LE.P	(cosx + sinx) ≤p
IF(X.LT. 0)	x = 0

Раздел 4. Операторы и графические функции

Тема 4.1. Операторы

Основные операторы FORTRANa

Для того, чтобы программа работала в любой версии FORTRANa рекомендуется заканчивать программу следующими операторами:

```
        STOP
        END
```

а подпрограмму:

```
        RETURN
        END
```

Оператор END может встречаться в программе только один раз, и всегда является последним оператором. Операторы STOP и RETURN – могут быть в любом месте программы и сколько угодно раз.

```
…….
…….
IF(X.EQ.0)   RETURN
……..
……..
RETURN
END
```

Оператор описания

Общая форма оператора:

x = e

x – переменная или элемент массива; e - константа, переменная или выражение.

x = 5.1

Тип x должен соответствовать типу переменной e. Иначе происходит преобразование типа e к типу x.

```
INTEGER A
          A =5.7        PRINT A          5
X = A +3 * B + C
```

Все переменные A, B, C должны быть определены до выполнения оператора присвоения.

Операторы управления

Операторы управления предназначены для изменения последовательности выполнения операторов, организации циклов и создания логической структуры программы.

GO TO , IF , DO , CONTINUE , PAUSE , STOP ,
CALL , RETURN , END

Оператор безусловного перехода GO TO

GO TO x

x – метка выполняемого оператора.

Безусловный оператор GO TO передаёт управление оператору с меткой x, а не следующему по порядку за оператором GO TO. Первый выполняемый оператор GO TO должен иметь метку.

```
……..
GO TO
1 X=0
3 X=10
```

Если нет метки 1, то оператор X=0 никогда не будет выполняться.

Вычислительный оператор GO TO

GO TO (x_1, x_2, …….. x_n)

x- метка выполняемого оператора;
i- выражение целого типа.

В результате выполнения оператора вычисляется значение выражения i и происходит передача управления оператору с меткой $x_к$, порядковый номер которой в списке меток совпадает с значением выражения i.

```
GO TO (1, 3, 5, 6)  N+1
A = 1
……..
1 A = 5
……
   3 A = N**3
……..
5 A = 0
```

Выражение N+1 может принимать значения 1 ≤N+1≤ 4. В противном случае выполняется оператор, следующий за оператором GO TO. Для удобства рекомендуется номера меток располагать по порядку номеров, а i– простая переменного целого типа

```
K = N+1
GO TO (1, 2, 3, 4)  K
A =1
…..
```

Назначенный оператор GO TO

GO TO m,(x_1, x_2, …….. x_n)

x- метка оператора , которому передаётся управление;
m – целая переменная, которой с помощью оператора ASSIGN должно быть присвоено значение одной из меток x_i.

```
ASSIGN  10  TO m
…….
ASSIGN  5  TO m
…….
GO TO m, (5, 10, 1)
…….
5 A = 1
……..
10 A = 4
……
```

В результате выполнения оператора GO TO происходит передача управления оператору с меткой, значение которой присвоено переменной m. Переменной m нельзя присвоить значение с помощью оператора присвоения:

m = 5 – неверно.

Замечание: назначенный оператор GO TO имеет более эффективный объектный код, чем вычисляемый оператор GO TO. Поэтому рекомендуется использовать этот оператор. Эти рекомендации даются в книгах по FORTRANy. Однако на практике они не всегда верны. Это связано с тем, что время выполнения оператора GO TO (любого) составляет ничтожную долю по сравнению с другими операторами. Зато он является более сложным, и соответственно менее удобным. Программы с использованием вычисляемого оператора GO TO более компактны и наглядны.

В целом рекомендуется, как можно меньше использовать операторы GO TO.

Операторы условного перехода IF

Предназначены для передачи управления в зависимости от выполнения некоторого условия.

Арифметический оператор IF:

IF(a) x_1, x_2, x_3

a – арифметическое выражение;
x_1, x_2, x_3 –метки выполняемых операторов.
При выполнении оператора вычисляется значение выражения a и производится передача управления оператору с меткой x_1, если значение выражения a<0, оператору с меткой x_2, если a≠0, и оператору с меткой x_3, если a>0.

$a < 0 \rightarrow x_1$
$a = 0 \rightarrow x_2$
$a > 0 \rightarrow x_3$

```
A=B*C
IF (A) 1, 3, 5
1 …….
3 …….
5 …….
```

Замечания: для перехода по нулю не рекомендуется использовать выражение вещественного типа, т.к. погрешность его вычисления может повлиять на сравнение с константой 0.0.

Логический оператор IF

IF (a) s

a- логическое выражение;

s- любой выполняемый оператор, кроме DO, IF, END

```
IF (X.EQ.0) GO TO 10
IF (X.EQ.1) A = X
B .=.X.**3
10 ……
  ……..
```

При выполнении оператора вычисляется логическое выражение и если оно «истинно», то выполняется оператор s. Если «ложь», то выполняется следующий оператор. Если s не оператор перехода любого типа, то после его выполнения выполняется оператор, следующий за IF.

Замечание: для минимизации времени выполнения логического оператора IF операнды выражения а необходимо размещать в следующем порядке:

1. если используется логическое выражение сравнения .OR. (или), то вначале располагаются условия, выполнения которых наиболее вероятно:

```
……
IF (A.LT.B. OR. A.LT.C. OR. A.LT.D) GO TO 3
…….
```

Переход на метку 3 будет выполняться, если справедливо хотя бы одно из трёх условий. При этом если, A < B, то остальные условия проверяться не будут;

2. если используется логическое выражение сравнения .AND., то вначале располагаются условия, которые выполняются реже всего.

```
IF (A.GT.B. AND. C. EQ.D) GO TO 3
```

Если A ≤ B, то следующее условие не проверяется.

Блочная форма оператора IF

Поддерживается только в языке FORTRAN-77 и предназначена для написания наглядных структурных программ. С помощью этого оператора реализуется. С помощью этого оператора реализуется выбор одной альтернативы из большого числа возможных. При этом исчезает необходимость в использовании операторов GO TO. Блочная форма образуется с использованием 4 операторов.

```
1). IF (a) THEN      } IF- блок
    …..…….
2). ELSE IF(a) THEN  } ELSE IF -блок
    ……..……..
3). ELSE             } ELSE -блок
    ……
    4). END IF
```

IF- группа

Из них обязательно должны присутствовать первый и четвёртый оператора. При выполнении блочного оператора IF выполняется значение выражения а. Если оно «истинно», то выполняются оператором IF- блока. Таким образом, называется последовательность операторов, расположенных между оператором IF соответствующим оператором ELSE IF, ELSE, END IF. После выполнения последнего оператора IF- блока управление передаётся оператору END IF и происходит выход из IF-группы.

Если значения выражения а «ложно», то IF- блок игнорируется, и управление сразу передаётся соответствующему оператору ELSE IF, ELSE, END IF.

Пример: определить состояние переменной I в зависимости от состояния переменной k следующим образом: I = -1 при k<0, иначе I = const остаётся прежним

```
IF (k. LT. 0) THEN  } IF- блок
    I = 1           }
    END IF
```

IF -группа

Оператор ELSE

Оператор ELSE никаких действий не выполняет и используется для выделения ELSE - блока. Таким образом, считается последовательность операторов между оператором ELSE и оператором END IF.

Операторы, входящие в ELSE – блока выполняются, если логическое выражение а в операторе IF или ELSE IF имеет значение «ложь».

Пример: пусть I = -1, при k < 0
I = 1, при k ≥ 0

```
IF (k. LT. 0) THEN  } IF- блок   } IF- группа
I = - 1
 ELSE               } ELSE -  блок
I = 1
END IF
```

С помощью оператора GO TO это выглядит следующим образом:

```
 IF (k. LT. 0) GO TO 3
I =  1
GO TO 5
3   I = - 1
5 CONTINUE
```

Такая программа менее наглядна.

Оператор ELSE IF

При выполнении оператора ELSE IF вычисляется значение логического выражения а и если оно «истинно», то выполняются операторы ELSE IF- блока. Таким блоком называется последовательность операторов между оператором ELSE IF и соответствующим оператором ELSE IF, ELSE, END IF. После выполнения операторов ELSE IF- блока управление передаётся соответствующему оператору END IF и происходит выход из всей IF- группы.

Если значение логического выражения а «ложно», то ELSE IF- блок игнорируется, управление сразу передаётся соответствующему оператору ELSE IF, ELSE, END IF.

Пример: пусть I = -1, при k < 0
I = 0, при k = 0
I = 1, при k ≥ 0

```
IF (k. LT. 0) THEN          } IF- блок
I = - 1
ELSE IF(k. EQ. 0) THEN      } ELSE IF- блок   } IF- группа
I = 0
 ELSE                       } ELSE - блок
I = 1
END IF
```

Оператор ELSE IF внутри IF- группы может быть несколько.

Оператор END IF

Оператор не выполняет никаких действий и используется в качестве ограничителя IF- группы в целом. Каждому блочному оператору IF должен соответствовать оператор END IF.

Вложенные IF – группы

IF – группы могут быть вложенными одна внутри другой. При этом операторы ELSE IF, ELSE, END IF относятся только к своей IF – группе. Это исключает пересечение различных IF – групп, в отличие от способов с использованием оператора GO TO.

I = -1, при k < 0
I = 0, при k = 0
I = 1, при k ≥ 0 J = -1 при k < 5, иначе J=0

```
 IF (k. LT. 0) THEN
I = - 1
IF (k. LT. -5) THEN
J = -1
ELSE
J=0
END IF
ELSE IF (k. EQ. 0) THEN
I = 0
ELSE
I = 1
END IF
```

IF- блок

IF- группа

Правила использования блочных операторов

1). Не допускается передача управления извне внутрь IF, ELSE IF, ELSE - блоков.
2). Нельзя передавать управление операторам ELSE IF, ELSE.
3). IF, ELSE IF и ELSE – блоки могут быть пустыми.

Рекомендации : пустые ELSE – блоки рекомендуется не опускать. При записи логического выражения в операторах IF, ELSE IF операнды размещать в таком же порядке, как и в логическом операторе IF.

Оператор CONTINUE

Общая форма x CONTINUE

x –метка оператора

Оператор указывает метку перехода и не выполняет действий связанных с вычислениями. Обычно используется как конечный оператор области цикла DO, для того чтобы избежать окончания цикла операторами GO TO, PAUSE, STOP, RETURN, а так же арифметическим или логическим IF, содержащим GO TO.

Пример: использования оператора CONTINUE, как метки перехода.

```
      IF (k. LT. 0) GO TO 3
      I =  1
      GO TO 5
3     I = - 1
5     CONTINUE
```

Использование CONTINUE в качестве конечного оператора цикла рассмотрим при изменении оператора DO.

Замечание: несмотря на то, что оператор CONTINUE не выполняет действий, связанных с вычислениями и не влияет на последовательность выполнения других операторов. Он может быть расположен в программе только выполняемых операторов.

Оператор DO

Оператор DO предназначен для организации циклов, т.е. для повторения некоторого участка программы заданное число раз. Общая форма оператора:

DO x i = m_1, m_2, m_3

x – метка последнего выполняемого оператора цикла (им не могут быть операторы GO TO, PAUSE, STOP, RETURN, IF);

i – переменная целого или вещественного типа, называемая переменная цикла;

m_1, m_2, m_3 - константы, переменные или выражения целого или вещественного типа:

m_1 – начальное значение переменной цикла i;

m_2 – её конечное значение;
m_3 – приращение переменной цикла i.

Примеры:

```
DO  3  I = n, m, k
DO  3  I = n+1, m/5, k*3
```

Оператор DO указывает на необходимость выполнить заданное количество раз операторы, входящие в цикл.

```
      REAL X(10)
      DO  3  I=1, 10
      x(I) = 0
    3 CONTINUE
```

Количество повторений цикла N определяется по формуле

$$N = INT\left(\frac{m_2 - m_1}{m_3}\right) + 1$$

INT - целая часть выражения $\frac{m_2 - m_1}{m_3}$;

Если N≤0 или $m_3 = 0$, то N = 0. процесс выполнения цикла в FORTRAN- 77 выглядит следующим образом:

1). Определяются значения m_1, m_2, m_3.
2). Определяется количество повторений цикла N и установление счётчика повторений.
3). Переменной i присваивается значение m_1
$i = m_1$
4). Добавление к переменной цикла i приращения m_3.
5). Проверка значений счётчика повторений на нуль.
6). Выполнение операторов цикла, если счётчик повторений >0.
7). Уменьшение счётчика на единицу.

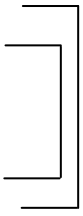
Пункты 4-7 выполняются многократно до тех пор, пока счётчик повторений цикла не станет равным нулю. После выполнения цикла i, переменная цикла сохраняет последнее значение:

```
      DO  3  I = 1, 10
      ……….
    3 CONTINUE
      WRITE (*,*) i

          1 1
```

Вложенные циклы

Внутри цикла могут находиться другие циклы DO, причём все операторы внутреннего цикла должны полностью размещаться внутри внешнего цикла, т.е. вложенные циклы не могут пересекаться.



Строки двумерного массива заполняются одинаковыми цифрами

```
      DO  3  I = 1, 10
      A(I) = B(i) / 10
       DO 5 J=1, 10
      C(I,J) = A(I)
    5 CONTINUE
3 CONTINUE
```

Внутренний и внешний циклы могут закрываться одним оператором

```
      DO  3  i = 1, 10
      A(i) = B(i) / 10
      DO 3 j=1, 10
      C(i,j) = A(i)
3  CONTINUE
```

Правила использования оператора DO

1). Параметры цикла i, m_1, m_2, m_3 не должны изменяться внутри цикла.

REAL A(30) DO 3 i = 1, 30	REAL A(30) DO 3 i = 1, 30
A(i) = 0 IF(I.EQ.10) I = I+10 3 CONTINUE	IF(I.GT.20) GO TO 5 A(i) = 0 3 CONTINUE

IF(J.GT.20) GO TO 5
IF(J.GT.10) J = J+10
A(J) = 0
3 CONTINUE
5 CONTINUE
	IF(I.GT.10) J = J+10
	A(J) = 0
	3 CONTINUE

Это пример присвоения нулевых значений элементам массива А с номерами от 1-10 и от 20-30.

2). Выход из цикла возможен в любом месте цикла. При этом переменная цикла определена и равна своему последнему значению.

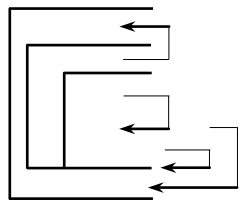
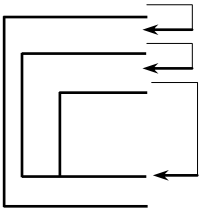
3). Из внутреннего цикла можно передавать управление внешнему циклу. Передача управления в обратном порядке не допускается.

4). Оператору, который является конечным оператором нескольких циклов, можно передавать управление только из самого внутреннего цикла.

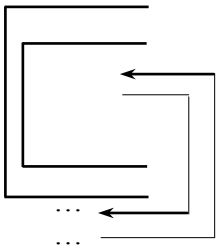
```
                DO    I = 1, 10
оставить место  IF(I.GT.5) GO TO 1      – неверно

                DO    J = 1, 10
                IF(J.GT.5) GO TO 1      - верно
                B(I,J) = 0
          1 CONTINUE
```

Организация переходов в циклах

	
верные переходы	неверные переходы

5). Оператор цикла DO может иметь расширенную область



…
…
…
… расширенная область

Расширенная область – это последовательность операторов, выполняемых между выходом из внутреннего цикла и возвратом в этот же цикл.

При использовании расширенной области должны выполняться следующие условия:

параметры цикла i, m_1, m_2, m_3 не должны изменяться в расширенной области;

расширенная область не может содержать другого оператора цикла DO, имеющего расширенную область.

6). Обращение к подпрограмме (и возврат из неё) допускается для любого цикла (внутреннего и внешнего), и любой расширенной области.

```
      DO 3 i =1, 10
      CALL TEST
    3 CONTINUE
```

7).Если блочный оператор IF находится внутри цикла, то оператор END IF также должен находится внутри цикла.

1 ÷ 50 = 0 50 ÷ 100 = 1 – заполнение массива

```
      REAL A (100)
      DO 3 I =1,100
      IF(I.LT.50) THEN
      A(I) =0
      ELSE
      A(I) =1
      END IF
  3 CONTINUE
```

```
      REAL A (100)
      DO 3 I =1,50
      A(I) =0
      A(I+50) =1
      END IF
    3 CONTINUE
```

8). Если оператор DO находится внутри IF–блока, ELSE IF -блока, ELSE -блока, то и последний оператор цикла должен быть внутри этого же блока. (массив обнуляется, если значение ключа =0, иначе присваивается 1)

```
      REAL A (100)
      KEY = X
      IF(KEY.EQ.0) THEN
      DO 3 I =1,100
      A(I) =0
  3 CONTINUE
      ELSE
      DO 5 I =1,100
      A(I) =1
  5 CONTINUE
      END IF
```

Замечание: в пятой версии языка FORTRAN фирмы Microsoft * допускается использование другой формы оператора цикла DO, которая очень удобна на практике. Общая форма

```
DO i =  m1, m2, m3
END DO
```

Здесь роль последнего оператора цикла играет оператор END DO.

```
      REAL A (100)
      DO i =  m1, m2, m3
      A(I) = 0
      END DO
```

Замечание: обычно в программах основная часть времени приходится на выполнение операторов цикла. Для сокращения этого времени существуют следующие рекомендации:

1). Развёртка цикла DO;
(один пример, мы уже приводили)

```
    REAL A(300), X(300)          REAL A(300), X(300)
    DO 3 I = 1, 300               DO 3 I = 3, 300, 3
    A(I) =X(I)                    A(I) =X(I)
3 CONTINUE                        A(I-1) =X(I-1)
                                  A(I-2) =X(I-2)
                                3 CONTINUE
```

2). Ссылки на элементы массива заменять ссылками на переменные. Произведение строк х

```
REAL  A (100), B(100), C(100)    REAL  A (100), B(100), C(100)
    DO I = 1,100                      DO I = 1,100
    DO J = 1,100                      x = 0
    A(I) = A(I) +B(J) +C(J)           DO J = 1,100
    END DO                            x = x + B(J) +C(J)
    END DO                            END DO
                                      A(I) = x
                                      END DO
```

3). Использование быстродействующих арифметических действий.

* вместо /
\+ вместо -

```
REAL  A (100)          REAL  A (100)
DO I = 1,100                  n = 1/3
A(I) = A(I)/3                 DO I = 1,100
END DO                        A(I) = A(I)*n
                              END DO
```

Неявная форма записи цикла

Чаще всего используется в операторах ввода-вывода. Пример: вывести на печать массив

```
INTEGER A (3,5)                INTEGER A (3,5)
DO i = 1,3                     DO i = 1,3
DO j = 1,5                     WRITE (1,3) (A(i,j), j=1,5)
WRITE (1,3) A(i,j)            END DO
END DO                      3  FORMAT (5I4)
END DO
3  FORMAT (5I4)
        INTEGER A (3,5)
         WRITE (1,3) ((A(i,j), j=1,5), i =1,3)
      3  FORMAT (5I4)
```

Тема 4.2. Графические функции

Структурный тип данных
Типы данных INREGER, REAL и др. являются простыми типами данных. В FORTRANe можно использовать структурные типы данных, состоящие из любой комбинации простых типов данных. Для объявления переменной структурного типа используется оператор:
RECORD /имя структуры/ имя переменной
например:
INREGER * 2 dymmy
RECORD /xycoord/ xy
RECORD /videoconfig/ screen

В примере оператор RECORD объявляет две переменные структурного типа: xy и screen; имена структур xycoord и videoconfig. Для задания содержимого самой структуры используется блочный оператор:

STRUCTURE /имя структуры/
…..
…...
…… простые типы данных; элементы структуры
…..
END STRUCTURE

Например:

```
STRUCTURE/xycoord/
INTEGER *2 xcoord
INTEGER *2 ycoord
END STRUCTURE
```

Для обращения к структурным переменным необходимо указать имя переменной и имя элемента структуры через точку:

k+xy.xcoord- координата по оси X текущей точки.

Оператор описания структуры STRUCTURE…END STRUCTURE должен предшествовать оператору RECORD, т.е. сначала описывается структура, а затем объявляются переменные данной структуры.

Замечания: 1). Элементом структуры может быть оператор RECORD, структура которого уже описана; 2). Имена переменных в структурах являются локальными, поэтому одно и то же имя может использоваться в разных структурах.

Каждая программа, использующая графическую библиотеку, должна явно объявлять любые процедуры. Для этого необходимо либо написать интерфейс для каждой процедуры, либо использовать файлы FGRAPH.FI, FGRAPH.FD. Файл FGRAPH.FI содержит объявления процедур в операторах INTERFACE; файл FGRAPH.FD содержит объявления структурных и символьных констант, операторы EXTERNAL для графических процедур. Совместно они обеспечивают доступ ко всем графическим процедурам. Эти файлы подключаются к программе с помощью оператора в начале программы:

```
INCLUDE 'FGRAPH.FI'
INCLUDE  'FGRAPH.FD'
REAL A, B, C
INTEGER Z
…
```

Если графические функции вызываются в подпрограммах, то в них подключаются только файлы FGRAPH.FD. Большинство графических процедур выполняется только в определённых графических режимах. Видеорежим обычно характеризуется следующими параметрами:

1. число пикселов по оси x;
2. число пикселов по оси y;
3. допустимое число столбцов текста;
4. допустимое число строк текста;
5. число цветов;
6. число размеров на пикселе;
7. число допустимых видеостраниц.

Для установки графического режима используется процедура setvideomode:

Эта процедура передаёт в оператор символьную константу, которая определяет какой режим следует установить. Если адаптер не поддерживает заданный режим, то процедура setvideomode возвращает 0. Тип функции INTEGER *2 (т.е. возвращаемое значение INTEGER *2).

Пример:

```
INCLUDE 'FGRAPH.FI'
INCLUDE  'FGRAPH.FD'
INTEGER*2 dummy
```

dummy = setvideomode ($MAXRESMODE) - выбирает максимально возможное разрешение (возвращает число строк);

dummy = setvideomode ($MAXCOLORMODE) - выбирает максимально возможное количество цветов (сколько, зависит от адаптера и монитора);
dummy = setvideomode ($VRES16COLOR) - 640*480, 16 цветов.
Для восстановления исходного режима используется переменная DEFAULTMODE.
dummy = setvideomode($DEFAULTMODE)
Числовые значения параметров видеорежима хранятся в структуре videoconfig. Поля структурной переменной videoconfig инициализируется при вызове процедуры getvideoconfig.
После установки видеорежима можно изображать графические объекты и текст.

Функция SETCOLOR

тип INTEGER *; вызов

dummy = setcolor (n)

где n– номер цвета, зависит от видеорежима.
В 16-ти цветном режиме цветам присвоены следующие номера:

0- чёрный	8- тёмно-серый
1- синий	9- светло-синий
2- зелёный	10- светло- зелёный
3- голубой	11- светло- голубой
4- красный	12- светло- красный
5- розовый	13- светло- розовый
6- коричневый	14- светло- коричне-вый
7- белый	15- ярко- белый

Процедура CLEASCREEN
- заполнение экрана текущим цветом фона (чистка экрана)

CALL clearcreen ($GCLEASCREEN)

Функция SETPIXEL

тип INTEGER *2, устанавливает элемент изображения в заданную область (рисует точку)
вызов:

dummy = setpixel (x, y)

где x, y - координаты

Процедура MOVETO

```
INTEGER *2 x, y
RECORD /xycoord/ xy
…
CALL MOVETO (x, y, xy)
```

Перемещение текущей позиции графического вывода в заданную точку.

Функция Lineto

Рисует линию с текущей позиции графического вывода до заданной позиции.
тип INTEGER *2; вызов

dummy = lineto (x, y)

где x, y- переменные типа INTEGER *2, содержащие координаты, куда надо провести линию.

Вывод текста

Для вывода текста можно использовать две функции:
SETTEXT position - задаёт место вывода текста;

OUTTEXT - выводи текст.
Пример:

RECORD /rc coord/ curpos

CALL SETTEXT position (10, 20, curpos) переменная curpos возвращает предыдущую текстовую позицию.

CALL OUTTEXT («ПРИМЕР »)

4.3. Лабораторные работы

Учебным планом не предусмотрено.

4.4. Практические занятия

№ п/п	*Номер раздела дисциплины*	*Тема практического занятия*	*Объем (час.)*	*Вид занятия в интерактивной, активной, инновационной формах, (час.)*
1	**2.**	Эквивалентные преобразования электрических схем	1	-
2	**3.**	Расчет потерь мощности в трансформаторах	1	-
3	**4.**	Расчёт характеристики мощности синхронного генератора	1	-
4	**4.**	Расчет статической устойчивости энергосистемы	1	-
5	**4.**	Расчёт электрической нагрузки жилых домов микрорайона	1	
6	**4.**	Определение удельного сопротивления кабеля	1	
		ИТОГО	**6**	5

4.5. Контрольные мероприятия: контрольная работа

Цель: изучение постановки задачи расчёта электрического режима сети.
Для выполнения контрольной работы студенту выдается индивидуальное задание, согласно которому требуется рассчитать рассчитать матрицу проводимостей схемы. На диагонали матрицы стоят собственные проводимости узлов. Собственная проводимость узла равна сумме проводимостей линий примыкающих к узлу. Внедиагональные элементы Zij равны проводимостям линий между узлами i и j. Проводимость линии равна величине обратной её сопротивлению. Составить систему уравнений, используя полученную матрицу и узловой ток в качестве свободного члена уравнения. Решить полученную систему и определить узловые напряжения. Исходные данные ввести из файла на диске. В файле указать количество линий, количество узлов, номера граничных узлов, сопротивления линий. Подписать обозначения параметров.
Расчет и изложение материала в работе следует выполнять в следующей последовательности:

1. Рассчитать проводимости линий для своего варианта и записать в таблицу параметров.
2. Рассчитать собственные проводимости узлов и записать в таблицу собственных проводимостей.
3. Составить полную матрицу проводимостей.
4. Разработать программу расчета полной матрицы проводимостей. Программа должна содержать: объявление переменных и массивов; открытие файла; ввод исходных данных; расчет собственных проводимостей узлов; расчет внедиагональных членов матрицы; печать полной матрицы проводимостей.
5. Подготовить файл исходных данных для расчета узловых токов.
6. Составить систему уравнений.

Решить систему уравнений. Для решения можно использовать MS EXCEL , либо вручную методом последовательного исключения переменных.

7. Разработать программу для решения системы уравнений из п.6.

Программа должна содержать: объявление переменных и массивов; ввод матрицы коэффициентов и массива свободных членов уравнений из файла; вызов стандартных под¬программ; печать таблицы результатов. Систему уравнений решить методом Гаусса. Имена стандартных подпрограмм реализующих данный метод: DECOMP (прямой ход метода Гаусса) и SOLVE (обратный ход метода Гаусса).

Рекомендуемый объем: работа выполняется на листах формата А4, объём 15 печатных страниц.

Выдача задания, прием РГР/кр/Р и защита КП (КР) проводится в соответствии с календарным учебным графиком

Оценка	Критерии оценки контрольной работы
Зачтено	«Зачтено» ставится при условии правильного выполнения всех заданий.
Не зачтено	Если не выполнено хотя бы одно из обязательных заданий.

5. МАТРИЦА СООТНЕСЕНИЯ РАЗДЕЛОВ УЧЕБНОЙ ДИСЦИПЛИНЫ К ФОРМИРУЕМЫМ В НИХ КОМПЕТЕНЦИЯМ И ОЦЕНКЕ РЕЗУЛЬТАТОВ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Компетенции / *№, наименование разделов дисциплины*	*Кол-во часов*	*Компетенции*		*Σ комп.*	$t_{ср}$, *час*	*Вид учебных занятий*	*Оценка результатов*
		ОПК-1	*ПК-2*				
1	2	3		5	6	7	8
1. Общие сведения и элементы языка	4	+	+	2	2	ЛК,СР	зачет
2. Структура программы	12	+	+	2	6	ЛК,ПЗ,КРз	зачет
3. Система в/в и встроенные функции	12	+	+	2	6	ЛК,ПЗ,КРз	зачет
4. Операторы и графические функции	80	+	+	2	40	ЛК,ПЗ,КРз	зачет
Всего часов	**108**	**54**	**54**	**2**	**54**		

6. ПЕРЕЧЕНЬ УЧЕБНО-МЕТОДИЧЕСКОГО ОБЕСПЕЧЕНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

1. Артемов И.Л. FORTRAN:основы программирования.И.Л. Артемов. – М.:Диалог-МИФИ, 2007.-304с.
2. Рыжиков Ю.И. Современный Фортран/ Ю.И.Рыжиков.- М.: Корона-Век, 2004.-288с.

7. ПЕРЕЧЕНЬ ОСНОВНОЙ И ДОПОЛНИТЕЛЬНОЙ ЛИТЕРАТУРЫ, НЕОБХОДИМОЙ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

№	*Наименование издания*	*Вид занятия*	*Количество экземпляров в библиотеке*, шт.	*Обеспеченность*, (экз./ чел.)
1	2	3	4	5
Основная литература				
1.	Артемов И.Л. FORTRAN:основы программирования/ И.Л.Артемов. – М.:Диалог-МИФИ.2007.- 304с.	Лк, ПЗ	36	1
Дополнительная литература				
2.	Васильченко В.В. Программирование Windows-приложений на языке FORTRAN.Элементы управления и графика Windows./В.В.Васильченко.- М.:Диалог-МИФИ. 2006.-400с.	Лк, ПЗ	31	1

3.	Информатика : учебник / А. Б. Золотов, П. А. Акимов [и др.]. - Москва : АСВ, 2010. - 336 с.	Лк	25	1

8. ПЕРЕЧЕНЬ РЕСУРСОВ ИНФОРМАЦИОННО - ТЕЛЕКОММУНИКАЦИОННОЙ СЕТИ «ИНТЕРНЕТ» НЕОБХОДИМЫХ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

1.Электронный каталог библиотеки БрГУ
http://irbis.brstu.ru/CGI/irbis64r_15/cgiirbis_64.exe?LNG=&C21COM=F&I21DBN=BOOK&P21DBN=BOOK&S21CNR=&Z21ID=.
2. Электронная библиотека БрГУ
http://ecat.brstu.ru/catalog .
3. Электронно-библиотечная система «Университетская библиотека online»
http://biblioclub.ru .
4. Электронно-библиотечная система «Издательство «Лань»
http://e.lanbook.com .
5. Информационная система "Единое окно доступа к образовательным ресурсам"
http://window.edu.ru .
6. Научная электронная библиотека eLIBRARY.RU http://elibrary.ru .
7. Университетская информационная система РОССИЯ (УИС РОССИЯ)
https://uisrussia.msu.ru/ .
8. Национальная электронная библиотека НЭБ
http://xn--90ax2c.xn--p1ai/how-to-search /.

9. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

9.1. Методические указания для обучающихся по выполнению практических занятий

Практическое занятие №1

Эквивалентные преобразования электрических схем

Цель работы: ознакомление с основами программирования на языке *FORTRAN* на примере решения простейших электротехнических задач; приобретение навыков работы с компилятором *FORTRAN Microsoft corp*. ; изучение способов ввода и вывода данных.

Задание: перед выполнением работы необходимо ознакомиться с программой Norton Commander (NC), и текстовым редактором NC, вызываемый нажатием F4 (или Shift + F4 для создания нового файла). Создать файл данных ISX.TXT. Создать исходный файл LAB.FOR и набрать текст программы. Создать объектный файл LAB.OBJ и выполняемый файл LAB.EXE. Получить контрольные решения. Файлы LAB.OBJ и LAB.EXE создаются трансляцией исходного файла LAB.FOR по команде FL LAB.FOR

Порядок выполнения: помимо ознакомления с программированием арифметических выражений в работе ставится целью освоение операторов ввода-вывода. Для расчёта №1 исходные данные ввести по оператору присваивания; Для расчёта№2 исходные данные ввести с клавиатуры; Для расчёта №3 и №4 исходные данные ввести с диска из файла *ISX.TXT*

Форма отчетности: результаты оформляются в форме отчета. В отчете должны содержаться следующие пункты: цель работы, исходные данные, текст программы, результаты расчета.

Основная литература : [1]

Дополнительная литература: [2]

Контрольные вопросы для самопроверки

1. Команды компиллятора
2. Операторы ввода-вывода.

Практическое занятие №2

Расчет потерь мощности в трансформаторах

Цель работы: ознакомление с программированием физических формул.

Задание: определить потери активной и реактивной мощности в трансформаторах, а также потери активной энергии за год.

Порядок выполнения: создать файл исходных данных; разработать текст программы;

скомпиллировать исполняемый файл; выполнить тестовый расчет.
Форма отчетности: результаты оформляются в форме отчета. В отчете должны содержаться следующие пункты: цель работы, исходные данные, текст программы, результаты расчета.
Основная литература : [1]
Дополнительная литература: [2]
Контрольные вопросы для самопроверки
1. Правила записи формул.
2. Операторы ввода-вывода.

Практическое занятие №3

Расчёт характеристики мощности синхронного генератора

Цель работы: Расчёт характеристики мощности синхронного генератора.
Задание: рассчитать характеристику электромагнитной мощности синхронного генератора $P = f(\delta)$ на интервале $\delta = [\ 0\ ;\ 3.14\ ]$ с шагом 0.314 (или $\delta = [\ 0\ ;\ 180\ ]$ с шагом 18), всего 11 точек . Результат представить в виде таблицы и графика.
Порядок выполнения: создать файл исходных данных; разработать текст программы; скомпиллировать исполняемый файл; выполнить тестовый расчет.
Форма отчетности: результаты оформляются в форме отчета. В отчете должны содержаться следующие пункты: цель работы, исходные данные, текст программы, результаты расчета.
Основная литература : [1]
Дополнительная литература: [2]
Контрольные вопросы для самопроверки
1. Ввод-вывод управляемый именованным списком.
2. Явная форма цикла.

Практическое занятие №4

Расчет статической устойчивости энергосистемы

Цель работы: выполнить анализ статической устойчивости энергосистемы путем расчета собственных значений (с.з.) матрицы коэффициентов системы линеаризованных уравнений.
Задание: написать основную программу, осуществляющую ввод матрицы коэффициентов, вызов стандартных подпрограмм и печать результатов расчета; результатом расчета должна быть таблица собственных значений (вещественной и мнимой частей) и вывод об устойчивости системы.
Порядок выполнения: создать файл исходных данных; разработать текст программы; скомпиллировать исполняемый файл; выполнить тестовый расчет.
Форма отчетности: результаты оформляются в форме отчета. В отчете должны содержаться следующие пункты: цель работы, исходные данные, текст программы, результаты расчета.
Основная литература : [1]
Дополнительная литература: [2]
Контрольные вопросы для самопроверки
1. Подпрограммы общего вида.
2. Передача данных через списки формальных и фактических параметров.

Практическое занятие №5

Расчёт электрической нагрузки жилых домов микрорайона

Цель работы: Ознакомление с методами определения расчетных нагрузок микрорайонов города.
Задание: представить справочники удельных нагрузок квартир и мощностей лифтов в виде одномерных массивов; определить нагрузку по отдельным типам домов; определить нагрузку всего микрорайона; определить реактивную и полную мощности.
Порядок выполнения: создать файл исходных данных; разработать текст программы; скомпиллировать исполняемый файл; выполнить тестовый расчет.
Форма отчетности: результаты оформляются в форме отчета. В отчете должны содержаться следующие пункты: цель работы, исходные данные, текст программы, результаты расчета.
Основная литература : [1]
Дополнительная литература: [2]
Контрольные вопросы для самопроверки
1. Скалярные действия над массивами.
2. Вложенные циклы.

Практическое занятие №6

Определение удельного сопротивления кабеля

Цель работы: изучение методов создания диалоговых программ, работа с таблицами данных.

Задание: создать файл исходных данных, содержащий только удельные сопротивления в зависимости от сечения и напряжения кабеля; разработать программу, которая объявляет и заполняет справочники сечений и напряжений, вводит таблицу удельных сопротивлений, вводит с клавиатуры значения расчетного напряжения и расчетного сечения кабеля, выполняет поиск удельного сопротивления кабеля в таблице, выводит найденное сопротивление в файл и на экран; программа должна определять удельное сопротивление для любого числа сечений и напряжений, т.е. ввод с клавиатуры должен выполняться многократно.

Порядок выполнения: создать файл исходных данных; разработать текст программы; скомпиллировать исполняемый файл; выполнить тестовый расчет.

Форма отчетности: результаты оформляются в форме отчета. В отчете должны содержаться следующие пункты: цель работы, исходные данные, текст программы, результаты расчета.

Основная литература : [1]

Дополнительная литература: [2]

Контрольные вопросы для самопроверки

1. Присвоение начальных значений.
2. Форматирование данных.

9.2. Методические указания по выполнению контрольной работы

Выполнение контрольной работы является важнейшей формой учебной работы и способствует закреплению и углублению знаний, полученных обучающимися на лекциях и практических занятиях, воспитывает навыки самостоятельной творческой работы и ведения инженерных расчетов.

Данные методические указания призваны помочь обучающимся выполнить расчет электрического режима электрической сети. Конкретные решения обучающийся принимает самостоятельно, основываясь на своих знаниях дисциплины.

В приложении даны все необходимые для выполнения работы справочные данные.

Методические указания по выполнению контрольной работы можно найти:

1. Алгоритмизация задач электроэнергетики: Методические указания к выполнению курсового проекта / Сост. А.А. Стародубцев. - Братск: БрГТУ, 2002. - 31 стр.

10. ПЕРЕЧЕНЬ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, ИСПОЛЬЗУЕМЫХ ПРИ ОСУЩЕСТВЛЕНИИ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ

- ОС Windows 7 Professional
- Microsoft Office 2007 Russian Academic OPEN No Level
- Антивирусное программное обеспечение Kaspersky Security.
- FORTRAN 95/2003/2008
- ПО "Антиплагиат"

11. ОПИСАНИЕ МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЙ БАЗЫ, НЕОБХОДИМОЙ ДЛЯ ОСУЩЕСТВЛЕНИЯ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ

Вид занятия	*Наименование аудитории*	*Перечень основного оборудования*	*№ ПЗ , №Лк*
1	2	3	4
Лк	Лекционная аудитория	-	№№1-9
ПЗ	Дисплейный	Оборудование	№№ 1-18

	класс	Интерактивная доска SMART Board 680I со встроенным XGA проектором Unifi 35 (77"/195,6 см); 16-ПК: CPU 5000/RAM 2Gb/HDD; Монитор TFT 19 LG1953S-SF; Принтер: HP LaserJet P3015; Сканер: EPSON GT1500	
СР	Читальный зал №3 (СР)	Оборудование 15 ПК- CPU 5000/RAM 2Gb/HDD (Монитор TFT 19 LG 1953S-SF);принтер HP LaserJet P3005	-
КРз	Читальный зал №3 (СР)	Оборудование 15 ПК- CPU 5000/RAM 2Gb/HDD (Монитор TFT 19 LG 1953S-SF);принтер HP LaserJet P3005	-

Приложение 1

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ПРОВЕДЕНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

1. Описание фонда оценочных средств (паспорт)

№ компетенции	Элемент компетенции	Раздел	Тема	ФОС
ОПК-1	Способность осуществлять поиск, хранение, обработку и анализ информации из различных источников и баз данных, представлять ее в требуемом формате с использованием информационных, компьютерных и сетевых технологий	**1.** Общие сведения и элементы языка	1.1. Общие сведения	Вопросы 1 -4
			1.2. Элементы языка	Вопросы 5 - 10
		2. Структура программы	2.1. Правила записи программы	Вопросы 11 – 18
			2.2. Процедуры. Порядок операторов	Вопросы 19 – 25
		3. Система в/в и встроенные функции	3.1. Система ввода/вывода	Вопросы 25 – 30
			3.2. Встроенные функции	Вопросы 30 – 34
ПК-2	Способность обрабатывать результаты эксперимента	**4.** Операторы и графические функции	4.1. Операторы	Вопросы 35 – 38
			4.2. Графические функции	Вопросы 39 - 42

2. Экзаменационные вопросы

№ п/п	Компетенции		ЭКЗАМЕНАЦИОННЫЕ ВОПРОСЫ	№ и наименование раздела
	Код	Определение		
1	2	3	4	5
1.	ОПК-1	Способность осуществлять поиск, хранение, обработку и анализ информации из различных источников и баз данных, представлять ее в требуемом формате с использованием информационных, компьютерных и сетевых технологий	1. Области применения языка программирования.	**1.** Общие сведения и элементы языка
			2. Достоинства языка программирования.	
			3. Существующие стандарты. Версии языка программирования.	
			4. Этапы разработки программы. Понятие исходного файла, объектного файла, исполняемого файла.	
			5. Правила записи программ (исходного файла). Порядок следования операторов в программе. правила записи одной строки.	**2.** Структура программы
			6. Основные конструкции языка программирования.	
			7. Какие символы можно использовать при записи исходного текста программ.	
			8. Типы констант. Типы переменных. Понятие стандартной и нестандартной длины переменной. Структурный тип данных	
			9. Способы объявления типов переменных. Массивы. Правила записи, способы объявления.	
			10. Порядок расположения элементов массива в операционной памяти.	
			11. Стандартные функции. Основные тригонометрические функции, обратные тригонометрические функции, корень квадратный, модуль числа, определение конца файла	**3.** Система в/в и встроенные функции

			12. Операторы ввода-вывода данных. Использование операторов ввода-вывода и именованных списков переменных.	
			13. Открытие файлов. Зарезервированные имена файлов.	
			14. Разделители форматов. Взаимодействие оператора FORMAT со списком ввода-вывода	
			15. Типы файлов. Открытие файлов прямо и последовательного доступа. Умолчания. Типы записей.	
			16. Операторы управления файлами.	
			17. Особенности арифметических действий под целыми числами.	
			18. Основные логические выражения отношения: больше, меньше, больше или равно, меньше или равно, равно, неравно.	
			19. Основные выполняемые операторы.	**4.** Операторы и графические функции
			20. Преобразование данных из внутреннего представления во внешнее. Понятие формата. Соответствие типов переменных и спецификаций форматов. Особенность спецификации формата X.	
			21. Операторы безусловного перехода GOTO.	
			22. Вычисляемый оператор GOTO. Назначенный оператор GOTO.	
			23. Арифметический оператор условного перехода IF. Особенность перехода по нулю.	
			24. Логический оператор условного перехода IF. Способы ускорения его работы.	
			25. Блочный оператор IF. Правила использования блочных операторов.	
			26. Первая форма записи циклов. Формула определения количества повторений цикла. Порядок выполнения цикла.	
			27. Вложенные циклы. Правила использования циклов. Схема правильных и неправильных переходов в циклах.	
			28. Вторая форма записи цикла. Способы ускорения работы циклов.	
			29. Неявные формы записи циклов. Печать таблицы из двух столбцов.	
			30. Циклы с логическим условием.	
			31. Присвоение начальных значений переменным и массивам с помощью оператора DATA.	
			32. Классификации подпрограмм.	
			33. Подпрограммы типа FUNCTION.	
			34. Подпрограммы типа SUBROUTINE.	
2.	ПК-2	Способность обрабатывать результаты эксперимента	35. Правило соответствия фактических и формальных параметров при передаче данных между подпрограммами через список параметров.	
			36. Правило соответствия между размерами фактического и формального массива. Способы описания формальных массивов в подпрограммах.	
			37. Подпрограммы типа оператор-функция.	
			38. Использование общих областей для передачи данных между подпрограммами. Преимущества этого способа.	
			39. Дополнительные точки входа в подпро-	

			граммы.	
			40. Назначение оператора INCLUDE.	
			41. Метакоманды. Стандартная и свободная форма записи исходного файла.	
			42. Атрибуты. Динамическое выделение памяти под массивы. Освобождение памяти.	

3. Описание показателей и критериев оценивания компетенций

Показатели	Оценка	Критерии
Знать (ОПК-1): - способы использования компьютерных технологий в своей предметной области; (ПК-1): - методы обработки экспериментальных данных. **Уметь** (ОПК-1): - применять компьютерную технику в своей профессиональной деятельности, в работе над проектами электроэнергетических и электротехнических систем и отдельных их компонентов; (ПК-1): - проводить анализ экспериментальных данных. **Владеть** (ОПК-1): - средствами компьютерной техники и сетевых технологий в своей предметной области. (ПК-1): – навыками обработки экспериментальных данных.	зачтено	Обучающийся глубоко и прочно усвоил программный материал и демонстрирует: - всестороннее знание программного материала; - умение правильного применения основных положений программного материала; - владеет всеми навыками, полученными в ходе изучения программного материала.
	не зачтено	Обучающийся допустил существенные ошибки при ответе на вопросы, на дополнительные вопросы давал неправильные ответы; все вышеуказанные разделы не усвоены.

4. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и опыта деятельности

Дисциплина «Компьютерные технологии» направлена на ознакомление с основными методами, применяемыми в компьютерных технологиях; на получение теоретических знаний и практических навыков по компьютерным технологиям. Изучение дисциплины «Компьютерные технологии» предусматривает:

– лекции,
– практические занятия,
– контрольная работа,
– зачет.

В ходе освоения раздела 1 «Общие сведения и элементы языка» студенты должны уяснить:

- этапы разработки программы;
- создание исполняемых файлов;

В ходе освоения раздела 2 «Структура программы» студенты должны уяснить:

- правила записи программ;

- порядок следования операторов;

В ходе освоения раздела 3 «Система в/в и встроенные функции» студенты должны уяснить:

- систему ввода-вывода;
- встроенные функции;

В ходе освоения раздела 4 «Операторы и графические функции» студенты должны уяснить:

- основные операторы;
- работу с графическими функциями.

Необходимо овладеть навыками и умениями применения изученных методов для определения основных характеристик компьютерных технологий.

В процессе изучения дисциплины рекомендуется на первом этапе обратить внимание на вопросы, связанные с применением компьютерных технологий в электроэнергетике.

При подготовке к зачету рекомендуется особое внимание уделить вопросам надежности компьютерных технологий.

В процессе проведения практических занятий происходит закрепление знаний по основным свойствам компьютерных технологий.

Самостоятельную работу необходимо начинать с изучения теоретического материала.

В процессе консультации с преподавателем необходимо выяснить все непонятные моменты.

Работа с литературой является важнейшим элементом в получении знаний по дисциплине. Прежде всего, необходимо воспользоваться списком рекомендуемой по данной дисциплине литературы.

Выполнение контрольной работы является важнейшей формой учебной работы и способствует закреплению и углублению знаний, полученных обучающимися на лекциях и практических занятиях, воспитывает навыки самостоятельной творческой работы и ведения инженерных расчетов.

Предусмотрено проведение аудиторных занятий в интерактивной форме (в виде лекции-дискуссии, лекции с разбором конкретных ситуаций) в сочетании с внеаудиторной работой.

Приложение 2

АННОТАЦИЯ
рабочей программы дисциплины

Компьютерные технологии

1. Цель и задачи дисциплины

Целью изучения дисциплины является: формирование знаний о принципах организации и практической реализации компьютерных технологий в электроэнергетических системах

Задачами изучения дисциплины являются: усвоение студентами основных принципов создания, разработки, отладки и тестирования алгоритмов и программ.

2. Структура дисциплины

2.1 Распределение трудоемкости по отдельным видам учебных занятий, включая самостоятельную работу: Лк 4 ч; ПЗ 6 ч; СР 94 ч.

Общая трудоемкость дисциплины составляет 108 часа, 3 зачетных единиц

2.2 Основные разделы дисциплины:

1. Общие сведения и элементы языка;
2. Структура программы;
3. Система в/в и встроенные функции;
4. Операторы и графические функции;

3. Планируемые результаты обучения (перечень компетенций)

Процесс изучения дисциплины направлен на формирование следующей компетенции:

ОПК-1 - Способность осуществлять поиск, хранение, обработку и анализ информации из различных источников и баз данных, представлять ее в требуемом формате с использованием информационных, компьютерных и сетевых технологий.
ПК-2 - Способность обрабатывать результаты эксперимента.

4. Вид промежуточной аттестации: зачет

Приложение 3

Протокол о дополнениях и изменениях в рабочей программе
на 20___-20___ учебный год

1. В рабочую программу по дисциплине вносятся следующие дополнения:

_____

_____

2. В рабочую программу по дисциплине вносятся следующие изменения:

_____

_____

_____

Протокол заседания кафедры №____ от «___» _____ 20 __ __ г.,
(разработчик)

Заведующий кафедрой _____ _____
(подпись) *(Ф.И.О.)*

Программа составлена в соответствии с федеральным государственным образовательным стандартом высшего образования по направлению подготовки 13.03.02 Электроэнергетика и электротехника (уровень бакалавриата) от «3» сентября 2015 г. №955

для набора 2014 года: и учебным планом ФГБОУ ВО «БрГУ» для заочной формы обучения от «3» июля 2018г. №413

Программу составил:

Стародубцев А.А., доцент, к.т.н.,каф.ЭиЭ _____

Рабочая программа рассмотрена и утверждена на заседании кафедры ЭиЭ

от «28» декабря 2018 г., протокол №5

Заведующий кафедрой ЭиЭ _____ Ю.Н. Булатов

СОГЛАСОВАНО:

Заведующий выпускающей кафедрой _____ Ю.Н. Булатов

Директор библиотеки _____ Т.Ф. Сотник

Рабочая программа одобрена методической комиссией ФЭиА

от «28» декабря 2018 г., протокол №5

Председатель методической комиссии факультета _____ А.Д. Ульянов

СОГЛАСОВАНО:

Начальник
учебно-методического управления _____ Г.П. Нежевец

Регистрационный №_____