Министерство образования Республики Беларусь
БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
Кафедра «Электропривод и автоматизация промышленных установок и технологических комплексов»

Информатика

Лабораторный практикум. Часть 1

для студентов специальности «Автоматизированные электроприводы»

Учебное электронное издание

Минск 2012

УДК 004(076.5)
ББК 32.81
П 12

А в т о р ы :

С.Н. Павлович

Р е ц е н з е н т ы :

В.П. Беляев, доцент кафедры полиграфического оборудования и систем обработки информации БГТУ, канд. техн. наук, доцент;

А.А. Гончар, доцент кафедры «Электроснабжение» БНТУ канд.техн. наук, доцент

В практикуме приведены описания десяти лабораторных работ, каждая из которых содержит цель работы, постановку задачи, порядок выполнения работы, контрольные вопросы, требования к содержанию отчета и теоретические сведения. Практикум предназначен для студентов специальности 1-53 01 05 «Автоматизированные электроприводы»

Белорусский национальный технический университет
пр-т Независимости, 65, г. Минск, Республика Беларусь
Тел.(017)292-77-52 факс (017)292-91-37
Регистрационный № БНТУ/ФИТР46-33.2012

Оглавление

В в е д е н и е

В настоящее время в обучении все шире используется **модульный подход**, в основе которого лежат такие основные принципы как системность, структуризация и модульность. Понятие «**модуль**» в рамках одной учебной дисциплины означает **блок информации**, *включающий в себя логически завершенную одну, две или более единиц учебного материала*.

При модульном подходе обучения учебная программа дисциплины (или ее отдельные разделы, темы) представляются в виде определенного количества конкретных модулей, с которыми обучаемый может работать самостоятельно или под руководством преподавателя. При этом *каждый модуль – это автономный учебный материал, предназначенный для освоения некоторой элементарной единицы знаний или умений*. Для каждого модуля (или для каждой единицы учебного материала) формулируется цель и четкая постановка задачи, дополненные необходимыми теоретическими сведениями.

В данном лабораторном практикуме приведены описания следующих пяти модулей дисциплины «Информатика» из раздела «Основы алгоритмизации и программирования инженерных задач на алгоритмическом языке Паскаль»:

М1 – Алгоритмизация задач (лабораторная работа № 1);

М2 – Базовые элементы алгоритмического языка Паскаль (лабораторные работы № 2 и № 3);

М3 – Программирование линейных алгоритмов (лабораторные работы № 4 и № 5);

М4 – Программирование разветвляющихся алгоритмов (лабораторные работы № 6 и № 7);

М5 – Программирование циклических алгоритмов (лабораторные работы №№ 8 – 10).

Теоретические сведения к указанным модулям практикума оформлены в виде Приложений №№ 1 – 5 (номер Приложения соответствует номеру модуля).

Общие требования по проведению лабораторных работ

При выполнении лабораторных работ ***студент обязан***:

1. Пройти инструктаж по технике безопасности и соблюдению требований пожарной безопасности в лаборатории с последующей регистрацией в соответствующем журнале.
2. До прихода в лабораторию самостоятельно ознакомиться с целью предстоящей работы, постановкой задачи и теоретическими сведениями по ней.
3. Перед выполнением работы заранее подготовить форму отчета (предварительный протокол), в котором представить такие сведения:
   - номер и название работы;
   - цель работы;
   - схему алгоритма решения задачи;
   - таблицу идентификаторов;
   - текст исходной Паскаль-программы.

4. Выполнить лабораторную работу и заполнить предварительный протокол экспериментальными данными (распечаткой текста отлаженной программы и результатами счета по ней на ПК).
5. По завершении выполнения лабораторной работы показать результаты преподавателю, привести в порядок рабочее место и получить разрешение покинуть лабораторию.

Цикл лабораторных работ предусматривает их фронтальное (последовательное) выполнение. Студенты, не выполнившие предыдущую работу, допускаются к выполнению последующей работы только после отработки предыдущей. *В случае пропуска занятий по уважительной причине студенту необходимо отработать соответствующие лабораторные работы с другой группой при наличии свободного рабочего места и с разрешения преподавателя.*

При возникновении затруднений студентам следует обратиться к описанию предыдущих работ или к рекомендованной литературе.

МОДУЛЬ М1 – «АЛГОРИТМИЗАЦИЯ ИНЖЕНЕРНЫХ ЗАДАЧ»

Лабораторная работа № 1

«АЛГОРИТМИЗАЦИЯ ИНЖЕНЕРНЫХ ЗАДАЧ»

Цель работы: *Ознакомление с методикой программирования и решения инженерных задач на персональном компьютере; изучение типовых алгоритмов, встречающихся при решении инженерных задач; приобретение практических навыков по разработке алгоритмов линейных, разветвляющихся и циклических вычислительных процессов в виде блок-схем (согласно ГОСТу 19.701-90).*

Постановка задачи

По варианту условия, определяемому номером бригады (из двух-трех студентов), разработать блок-схемы алгоритмов решения следующих задач (табл. 1.1):

Таблица 1.1

Варианты заданий

№ вариантов	Условия		
	Линейная структура	Разветвляющаяся структура	Циклическая структура
1	2	3	4
1	Вычислить объем усеченного конуса: $V=1/3\pi h\,(R^2+r^2+Rr)$	Вычислить значение: $\omega_1=\begin{cases} e_1^{2t}+\cos(xt_2) \text{ при } t_1<\sqrt{t_2+x};\\ \ln(xt_1)\cdot\sqrt{sint_2} \text{ при } t_1=\sqrt{t_2+x};\\ xt_1+tg(t_2) \text{ при } t_1>\sqrt{t_2+x} \end{cases}$	Вычислить значения: $\beta_i=\dfrac{ax_i^2+e_i^{2Z}}{t\cdot\sqrt{sinz_i}};$ $i=\overline{1,n};\ n\le 4$
2	Вычислить значение функции $y=ae^{-bx}+\sin(\omega t)$	Вычислить значение: $\beta_2=\begin{cases} 3q+\sqrt{\lvert x^3\cdot\sin(z)\rvert} \text{ при } \sin(z)<q;\\ \sqrt{aq}+e^{\alpha z} \text{ при } \sin(z)=q;\\ \lvert\alpha^5\rvert\cdot\ln\sqrt{\ln\cos(z)} \text{ при } \sin(z)>q \end{cases}$	Вычислить значения: $\alpha_i=\dfrac{z_i+\sqrt[3]{q^2}}{b\cdot\ln^2\omega_i};$ $i=\overline{1,n};\ n\le 5$

8 Вычислить значение функции:

$$y=x^{a}+e^{\sin(\omega t)} \qquad k=\begin{cases} \ln|f|+|g|, \text{если } |f\cdot g|>0\\ e^{f+g}, \text{если } |f\cdot g|<0\\ f+g, \text{если } |f\cdot g|=0 \end{cases} \qquad t_i=\frac{z_i+\sin^2 g}{2tg(z_i+g)};\quad i=1,n;\ n\le 5$$

Окончание табл. 1.1

1	2	3	4
3	Вычислить площадь треугольника $S = \sqrt{p(p-a)(p-b)(p-c)}$, где p = (a+b+c)/2	Вычислить значение: $\gamma_1 = \begin{cases} \alpha^3 + \sqrt{\cos(y^2)} \text{ при } y < \ln\beta; \\ \lg(x+\omega)^2 + \lvert y^3 \rvert \text{ при } y = \ln\beta; \\ \sqrt{\operatorname{tg}(y^5)} + x\omega \text{ при } y > \ln\beta \end{cases}$	Вычислить значения: $\gamma_i = \frac{3q_i\sqrt{y_i^2 + \cos(z)}}{2\operatorname{tg}^2(b+q_i)}$; $i = \overline{1,n};\ n \le 5$
4	Вычислить площадь правильного n-угольника $S = 1/2 \cdot n \cdot R^2 \cdot \sin\alpha$	Вычислить значение: $\tau_5 = \begin{cases} e_1^{2T} + \cos(xr_2) \text{ при } r_1 < r_2 x; \\ \ln(xr_1) + \sin(r_2) \text{ при } r_1 = r_2 x; \\ xr_1 + \operatorname{tg}(r_2) \text{ при } r_1 > r_2 x \end{cases}$	Вычислить значения: $\omega_i = \frac{\arcsin(z_i^2) + z_1}{\cos z_1 + \lvert z_1^5 \rvert}$; $i = \overline{1,n};\ n \le 5$
5	Вычислить площадь сектора с углом α° $s_c = \pi \cdot R^2 \cdot \alpha/360$	Вычислить значение: $\sigma_6 = \begin{cases} 3\lvert q + y^3 + \sin(z) \rvert \text{ при } \sin(z) < q; \\ \sqrt{aq} + e^{\alpha z} \text{ при } \sin(z) = q; \\ \lvert \beta^3 \rvert \cdot \ln\cos(z) \text{ при } \sin(z) > q \end{cases}$	Вычислить значения: $\omega_i = \frac{z_i + \sin q^2}{t \cdot \ln^2 \alpha_i}$; $i = \overline{1,n};\ n \le 5$
6	Вычислить полную поверхность цилиндра $S_п = 2\pi R(H+R)$	Вычислить значение: $R_7 = \begin{cases} \alpha^3 + \cos(y^3) \text{ при } y < \ln\alpha; \\ \lg(x+\omega)^2 + \lvert y^5 \rvert \text{ при } y = \ln\alpha; \\ \operatorname{tg}(y^3) + e^{x\omega} \text{ при } y > \ln\alpha \end{cases}$	Вычислить значения: $t_i = \frac{3g_i x_i^2 + \cos(z)}{2\operatorname{tg}^2(t+g_i)}$; $i = \overline{1,n};\ n \le 5$
7	Вычислить длину хорды сегмента с центральным углом α $L = 2R \cdot \sin(\alpha/2)$	Вычислить значение: $f_8 = \begin{cases} \beta^3 + \sin(z^2) \text{ при } z < \ln\beta; \\ \lg(q+\omega)^2 + \lvert z^7 \rvert \text{ при } z = \ln\beta; \\ \operatorname{tg}(z^3) + e^{q\omega} \text{ при } z > \ln\beta \end{cases}$	Вычислить значения: $t_i = \frac{x_i y_i^2 + \ln(d)}{2\operatorname{tg}^2(t+x_i)}$; $i = \overline{1,n};\ n \le 5$

Содержание лабораторной работы

Лабораторная работа включает:

1. Ознакомление с теоретическим материалом (см. Приложение 1) .
2. Построение в отчете по лабораторной работе блок-схем алгоритмов согласно варианту задания.

Контрольные вопросы

1. Дать определение алгоритма.
2. Что Вы понимаете под термином «алгоритмизация задачи»?

3. Может ли задача иметь несколько алгоритмов решения?

4. На каких принципах основано построение схем алгоритмов?

5. Каковы особенности построения схем алгоритмов линейных, разветвляющихся и циклических вычислительных процессов?

Содержание отчета

Отчет по выполненной работе оформляется на основании варианта задания и должен содержать следующие сведения:

1. Номер и наименование лабораторной работы.
2. Цель работы.
3. Постановку задачи.
4. Решение задачи в виде блок-схем алгоритмов согласно варианту задания.
5. Выводы по работе.

Приложение 1 ***(к модулю М1)***

Теоретические сведения к лабораторной работе № 1- «Алгоритмизация инженерных задач»

1. МЕТОДИКА ПРОГРАММИРОВАНИЯ И РЕШЕНИЯ ИНЖЕНЕРНЫХ ЗАДАЧ НА ПК

Методика подготовки (программирования) и решения инженерных задач на ПК включает в себя выполнение следующих основных этапов:

1. Математическую формулировку задачи.
2. Выбор метода вычисления.
3. Разработку схемы алгоритма решения задачи.
4. Написание программы на алгоритмическом языке.
5. Ввод текста программы в память ПК и отладку ее.

6. Ввод исходных данных и выполнение вычислений по программе.

На ***первом этапе*** условие задачи записывается в виде последовательности формул, которые в дальнейшем будут использоваться в вычислениях.

На ***втором этапе*** выбирается такой метод решения математически сформулированной задачи, который позволяет свести поиск результата к выполнению последовательности элементарных математических операций. Для большинства практических задач разработаны численные методы, обеспечивающие приближенное решение с требуемой точностью.

На ***третьем этапе*** на основе выбранного метода разрабатывается схема алгоритма решения задачи (см. пункты **2** и **3).**

На ***четвертом этапе,*** используя схему алгоритма, составляют программу решения задачи на определенном алгоритмическом языке в виде последовательности соответствующих операторов.

На ***пятом этапе*** с помощью системы программирования вводят программу в память ПК, транслируют, редактируют, проверяют правильность ее написания и ввода.

На ***шестом этапе*** вводят исходные данные и выполняют вычисления по программе.

2. РАЗРАБОТКА АЛГОРИТМА РЕШАЕМОЙ ЗАДАЧИ

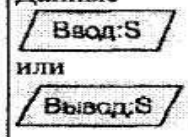
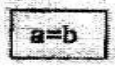
Под алгоритмом понимается определенная последовательность предписаний (инструкций, действий, операций) по переработке исходных и промежуточных данных в результат решения задачи, т.е. алгоритм представляет собой общую схему решения конкретной задачи. При разработке алгоритмов сложных задач выделяют составные части, различные по назначению. Каждая часть может представлять собой отдельный алгоритм. Иногда этот алгоритм можно разбить на ряд еще более простых алгоритмов и т.д. Глубина разработки алгоритма (детализация) зависит от наличия разработанных ра-

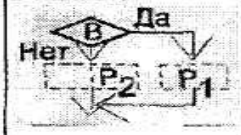
нее стандартных алгоритмов решения типовых задач и от условий решаемой задачи. В итоге алгоритм решения задачи сводится к определенной последовательности таких действий: начало решения, ввод данных, вычисления по формулам, обращение к подпрограммам (другим мини-программам), проверка условий, влияющих на ход вычислительного процесса, организация повторяющихся участков вычислений, вывод данных (исходных, промежуточных, результатов решения задачи), конец решения.

Из различных способов описания алгоритмов (словесный, операторный, схемный) наиболее распространен (из-за большей наглядности, облегчающий затем написание по алгоритму самой программы на каком-нибудь алгоритмическом языке) схемный способ, при котором алгоритм представляется в виде символов (блоков), выполняющих определенные действия, и связей между ними с помощью линий, указывающих очередность выполнения этих символов при вычислениях. Форму, размеры, наименование символов и выполняемые ими функции, а также правила построения схем алгоритмов определяет ГОСТ 19.701-90 [5]. В табл. 1.2 приведены основные, чаще употребляемые символы схем алгоритмов и форма их записи (программирование в общем формате) на алгоритмическом языке Паскаль.

Таблица 1.2

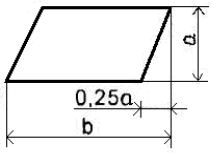
Некоторые символы и форма их записи на Паскале

Название и форма обозначения символа	*Выполняемая функция*	*Общая форма записи символа на Паскале*	*Примечание*
1	2	3	4
Терминатор 	Начало выполнения программы Конец программы	Заголовок программы, описательные разделы и операторная скобка Begin раздела операторов End.	
Данные 	Ввод данных Вывод данных	Read(S); или ReadLn(S); Write(S); или WriteLn(S);	S – список вводимых переменных S – список выводимых элементов
Процесс 	Выполнение вычислений по формуле	a:=b;	a – имя переменной; b – арифметическое или логическое выражение

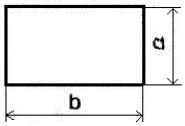
1	2	3	4
Решение 	Выбор дальнейших вычислений в зависимости от значения B	If B then P_1; или If B then P_1 else P_2;	B – логическое выражение P_1, P_2 – операторы (простые, составные или вложенные)
Подготовка 	Начало цикла с известным числом повторения и шагом изменения параметра цикла *i*, равным 1	For i:=n_1 to n_2 do p; или (если $n_1 > n_2$) For i:=n_1 downto n_2 do p;	i – параметр цикла n_1, n_2 – начальное и конечное значение i p – тело цикла (простой, составной или вложенный оператор)

Приведем условные обозначения чаще используемых символов (блоков) и некоторые правила выполнения схем алгоритмов из ГОСТа 19.701-90.

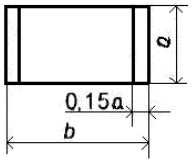
1. Данные. Символ отображает данные, носитель данных не определен (символы данных во многих случаях представляют способы ввода/вывода данных).

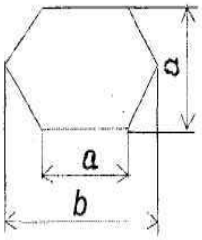


2. Процесс. Символ отображает обработку данных (вычисления по формулам).

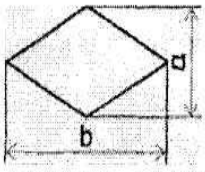


3. Предопределенный процесс. Символ отображает подпрограмму, модуль.

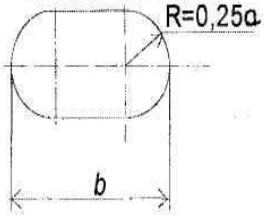




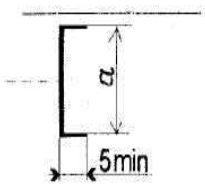
4. ***Подготовка.*** Символ отображает модификацию команды или группы команд с целью воздействия на некоторую последующую функцию (начало цикла с заданным числом повторения).



5. ***Решение.*** Символ отображает выбор направлений выполнения алгоритма в зависимости от некоторых условий.



6. ***Терминатор.*** Символ отображает выход во внешнюю среду и вход из внешней среды (начало или конец схемы программы), прерывание (остановка, пуск).



7. *Линия.* Поток данных или управления.

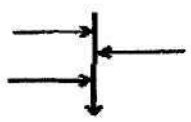
8. Комментарий. Символ отображает связь между элементом схемы и пояснением (используется, если текст внутри символа не помещается).

Символы (блоки) в схеме алгоритма должны быть расположены равномерно, быть по возможности одного размера; *не должны изменяться углы и другие параметры, влияющие на соответствующую форму символов* (размер *a* выбирается из ряда 10, 15, 20, ... мм, а размер b = 1,5*a* или 2*a* согласно ГОСТ 19.003-80).

Помещаемый внутри символа текст записывается слева направо и сверху вниз независимо от направления потока выполнения символов. Текст должен быть минимальным и достаточным для понимания выполняемых функций данного символа; если объем текста превышает размеры символа, то следует использовать символ комментария.

Для большей ясности на линиях потока в схеме алгоритма используются стрелки. Направление линий потока (данных или управления) слева направо и сверху вниз считается ***стандартным*** и стрелкой может не обозначаться. Линии, отличные от стандартного направления или имеющие изломы, должны обязательно обозначаться стрелками.

В схемах алгоритмов следует избегать пересечения линий. Пересекающиеся линии не имеют логической связи между собой, поэтому изменение направления в точках пересечения не допускается. Две или более входящих линий могут объединяться в одну исходящую линию, при этом место объединения должно быть смещено:



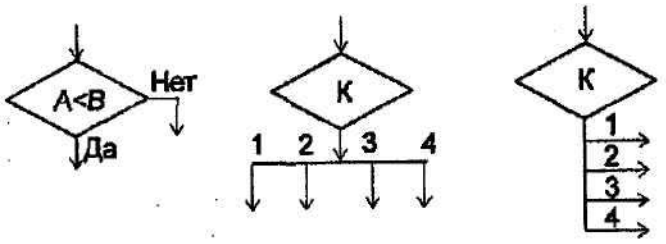
Линии в схемах алгоритмов должны подходить к символу либо слева, либо сверху, а исходить либо справа, либо снизу (по центру символа).

Несколько выходов из символа можно показывать:

-несколькими линиями от данного символа к другим символам;

-одной линией от данного символа, которая затем разветвляется на соответствующее число линий. Каждый выход из символа должен сопровождаться соответствующим значением условия разветвления.

Пример.



Символы в схеме алгоритма могут помечаться идентификаторами, например, для ссылки на них в других частях документации. Идентификатор располагается слева над символом.

При разработке схемы алгоритма и написании Паскаль-программы целесообразно использовать некоторые простые приемы, позволяющие уменьшить затраты времени на вычисления (при решении задачи по программе):

- вместо операции возведения в целую степень (для низких степеней) использовать операцию умножения (например, $a^3 = a \cdot a \cdot a$);

- выражение, вычисляемое в программе многократно с одними и теми же данными, необходимо вычислять один раз, присваивая его значение промежуточной переменной, используемой затем в дальнейших вычислениях как известной;

- в качестве границ параметров цикла использовать не выражения, а промежуточные переменные, вычисляемые до начала выполнения цикла;

- все повторяющиеся внутри цикла вычисления с одинаковыми данными выносить за пределы цикла, выполняя их один раз до начала цикла.

Рассмотрим в следующем разделе некоторые характерные приемы алгоритмизации типовых задач на конкретных примерах.

3. СТАНДАРТНЫЕ СХЕМЫ АЛГОРИТМОВ

3.1. Линейный алгоритм

Линейным называется алгоритм, в котором все блоки выполняются последовательно один за другим. При разработке алгоритма решаемой задачи следует выбирать наилучший вариант в смысле некоторого критерия, в качестве которого используют либо оценку точности решения задачи, либо затраты времени на решение, либо некоторые интегральные критерии.

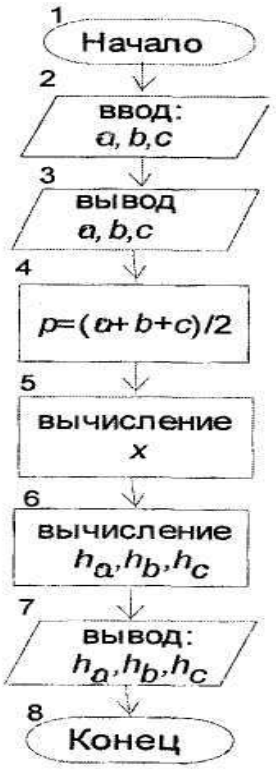


Рис. 1

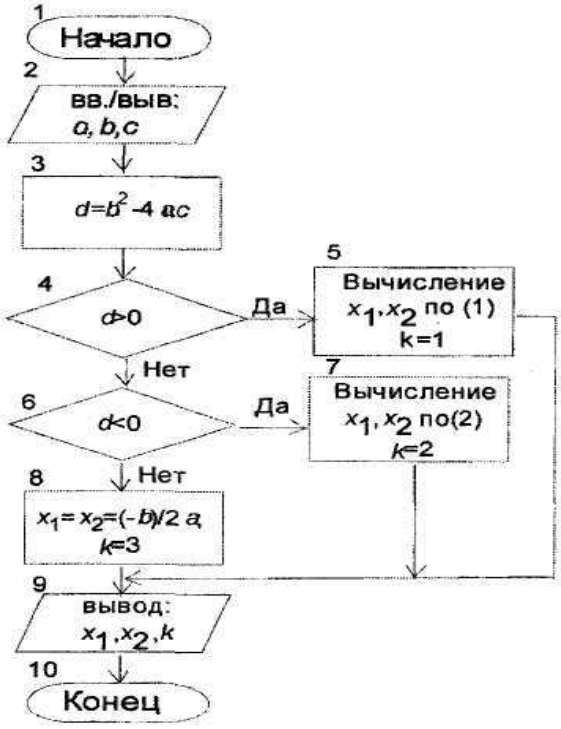


Рис. 2

Пример.

Разработаем линейный алгоритм для определения высот: h_a, h_b , h_c треугольника по заданным длинам его сторон *a, b, c:*

$h_a = 2/a\sqrt{p\cdot(p-a)\cdot(p-b)\cdot(p-c)}$;

$h_b = 2/b\sqrt{p\cdot(p-a)\cdot(p-b)\cdot(p-c)}$;

$h_c = 2/c\sqrt{p\cdot(p-a)\cdot(p-b)\cdot(p-c)}$,

где $p = (a + b + c)/2$.

При решении данной задачи, с целью уменьшения затрат на вычисления, определять высоты будем не по приведенным выше формулам, а с использованием промежуточной переменной:

$$x = 2\sqrt{p\cdot(p-a)\cdot(p-b)\cdot(p-c)} .$$

Тогда $h_a = x/a$; $h_b = xlb$; $h_c = x/c$ и блок-схема алгоритма решения данной задачи будет иметь вид, представленный на рис. 1.

3.2. Разветвляющийся алгоритм

На практике часто, в зависимости от исходных данных или от результатов промежуточных вычислений, бывает необходимо организовать дальнейшие вычисления по одним или другим формулам, т.е. вычислительный процесс в зависимости от выполнения некоторых логических условий должен идти по нескольким направлениям (ветвям). Алгоритм такого вычислительного процесса называют ***разветвляющимся.***

Пример.

Разработаем разветвляющийся алгоритм вычисления корней квадратного уравнения

$ax^2 + bx + c = 0$ с вещественными коэффициентами *a, b , c.*

Алгоритм решения данной задачи (рис. 2) содержит три ветви (в зависимости от значения дискриминанта $d = b^2 - 4ac$):

1. Если $d > 0$, то корни вещественные и определяются по формулам:

$$x_1 = (-b + \sqrt{d}) / (2a); \; x_2 = (-b - \sqrt{d}) / (2a). \qquad (1)$$

2. Если $d < 0$, то корни комплексно-сопряженные вида $x_1 \pm jx_2$, а их вещественная (x_1) и мнимая (x_2) части вычисляются по формулам:

$$x_1 = -b/(2a); \quad x_2 = \sqrt{-d}/(2a). \qquad (2)$$

3. Если d = 0, то корни вещественные и равные между собой: $x_1 = x_2 = -b/(2a)$.

В данном алгоритме вид корней при выводе результатов вычисления определяется значением переменной ***к:*** если ***к*** = 1 , то корни вещественные и разные; если ***к*** = 2, то корни комплексно-сопряженные; если ***к*** = 3, то корни вещественные и равные между собой.

3.3. Циклические алгоритмы

Циклическим алгоритмом называют вычислительный процесс, в котором решение задачи или ее части сводится к многократному вычислению по одним и тем же формулам при различных значениях входящих в них некоторых переменных. Многократно повторяющиеся участки такого вычислительного процесса называются ***циклами.***

Различают циклы с заданным (известным) и неизвестным числом повторений. К последним относятся так называемые ***итерационные циклы,*** характеризующиеся последовательным приближением к искомому значению результата с заданной точностью (например, при вычислении суммы членов сходящегося бесконечного ряда чисел).

Переменную, изменяющуюся в цикле, называют ***параметром цикла.***

В одном цикле могут быть один или несколько параметров. *Цикл с несколькими одновременно изменяющимися параметрами* организуется по схеме

построения цикла с одним каким-либо параметром, а для остальных параметров перед началом цикла задаются их начальные значения, а внутри цикла вычисляются их текущие значения для очередных повторений цикла.

Пример 1.

Составим схему алгоритма по вычислению функции *S,* равной сумме членов y_i, т.е.

$$S = \sum_{i=1}^{20} y_i, \; y_i = x_i^2 /i; \; x_i\text{- элементы массива } (x_1, x_2, \ldots, x_{20}).$$

Эта задача относится к циклическому вычислительному процессу с заданным числом повторения цикла по накапливанию суммы членов y_i .

Схема алгоритма решения данного примера представлена на рис. 3, в котором блок 3 задает начальное значение суммы *S* перед циклом, а блок 5 вычисляет значение слагаемых y_i и накапливает искомую сумму.

Примечание. Алгоритм по вычислению функции *P*, равной произведению членов y_i , т.е.

$$P = \Pi\left(x_i^2 /i\right)_{i=1}^{20}$$

аналогичен алгоритму на рис. 3 с той лишь разницей, что для накапливания произведения будет использоваться в блоке 5 формула $P = P \bullet y_i$, а началь-

ное значение произведения P в блоке 3 должно быть равно единице.

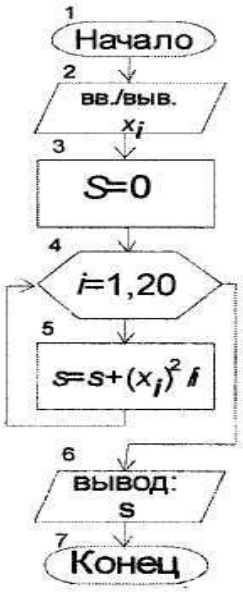


Рис. 3

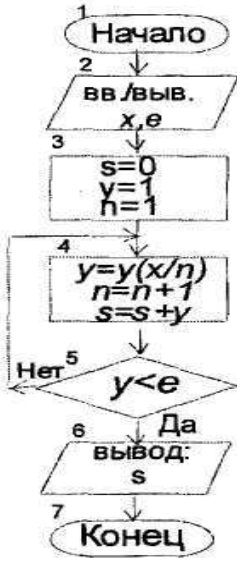


Рис. 4

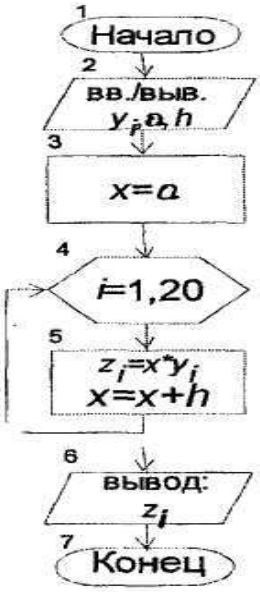


Рис. 5

Пример 2.

Разработаем алгоритм по вычислению суммы членов сходящегося бесконечного ряда чисел:

$$S= x + x^2/2! + x^3/3! + ... = \sum_{n=1}^{\infty}(x^n/n!)$$

с точностью до члена ряда, меньшего ***e***.

Здесь имеет место итерационный цикл, так как заранее не известно, при каком ***n*** выполняется условие $(x^n/n!) < e$.

Сравнивая два соседних члена ряда, видим, что $y_n / y_{n-1} = x / n$.

Поэтому для уменьшения затрат времени на вычисление текущего члена ряда целесообразно в цикле использовать *рекуррентную* формулу: $y_n = y_{n-1} \cdot x/n$, т.е. формулу, использующую при вычислении члена y_n уже вычисленное значение предыдущего члена y_{n-1}. А чтобы использовать эту формулу для вычисления первого члена ряда $y_1 = y_0 \cdot x/1$, необходимо положить начальное

значение $y_0 = 1$. Тогда схема алгоритма решения данного примера будет иметь вид, представленный на рис. 4.

Пример 3.

Разработаем алгоритм по вычислению функции $z_i = x \cdot y_1$, если x изменяется одновременно с i от начального значения a с шагом h, а y_t - элементы массива $(y_1, y_2, ..., y_{20})$.

Здесь в цикле, повторяемом 20 раз, изменяются одновременно 2 параметра: простая переменная x и i - индекс массива y. Схема алгоритма решения данной задачи представлена на рис. 5, где блок 4 задает закон изменения параметра i, блок 3 (перед циклом) - начальное значение параметра x, а блок 5 (внутри цикла) вычисляет (кроме z_i) новое значение параметра x для очередного выполнения цикла.

Пример 4.

Разработаем алгоритм по вычислению суммы положительных элементов каждой строки матрицы $a\ (10 \bullet 15)$.

Для вычисления суммы элементов одной строки заданной матрицы необходимо организовать цикл с целью перебора всех элементов строки, поэтому параметром этого цикла следует выбрать номер столбца $к$. Перед циклом нужно задать начальное значение суммы $S_t = 0$.

Для вычисления суммы положительных элементов каждой строки матрицы необходимо организовать другой цикл (с параметром номера строки i), охватывающий первый цикл. Здесь мы имеем алгоритм со структурой вложенного цикла. ***Вложенным*** называется цикл, содержащий внутри себя один или несколько других циклов, при этом цикл, охватывающий другие циклы, называется ***внешним***, а остальные циклы - ***внутренними.*** Параметры этих циклов изменяются не одновременно, так как при одном каком-либо значении параметра внешнего цикла параметр внутреннего цикла принимает

по очереди все свои значения. Схема алгоритма данного примера приведена на рис. 6.

3.4. Вычисление полинома

Для вычисления полинома n-й степени

$$y = a_1x^n + a_2x^{n-1} + ... + a_nx + a_{n+1}$$

удобно использовать формулу Горнера:

$$y = ((a_1x + a_2)x + ... + a_n)x + a_{n+1}.$$

Если выражение в самых внутренних скобках обозначить y_i, то значение выражения в следующих скобках можно вычислять по рекуррентной формуле: $y_{i+1} = y_ix + a_{i+1}$, а значение полинома y получим после повторения этого процесса в цикле n раз. Начальное значение y_i целесообразно взять равным a_1, а цикл начать с i = 2. Тогда алгоритм для вычисления полинома *n-й* степени по формуле Горнера можно представить в виде схемы рис. 7. Все коэффициенты полинома сводятся в массив из n+1 элементов, при этом следует иметь в виду, что если полином не содержит членов с некоторыми степенями x, то в массиве коэффициентов a_i на соответствующих местах необходимо помещать коэффициенты, равные нулю.

3.5. Нахождение наибольшего или наименьшего значения функции

Нахождение наибольшего (или наименьшего) значения функции $y = f[x)$ выполняется по циклу, в котором вычисляется текущее значение функции и сравнивается с наибольшим (или с наименьшим) из всех предыдущих значений этой функции. Если текущее значение функции оказывается больше из наибольшего (или меньше из наименьшего) из предыдущих значений, то его считают новым наибольшим (или наименьшим) значением.

При этом в качестве начального значения y_{max} необходимо взять очень малое значение (например, число $-1 \cdot 10^{10}$), а в качестве начального значения y_{min} - очень большое число (например, $+ 1 \cdot 10^{10}$).

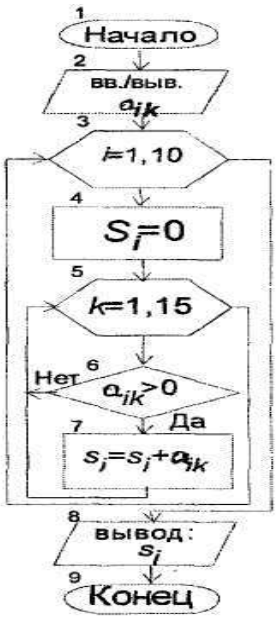


Рис. 6

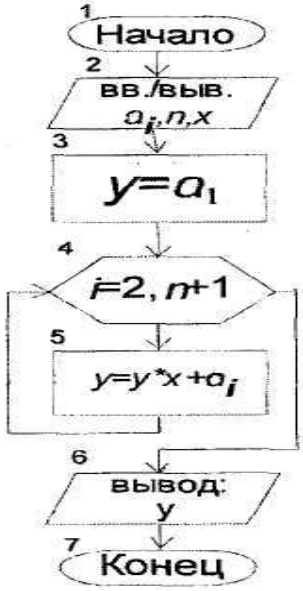


Рис. 7

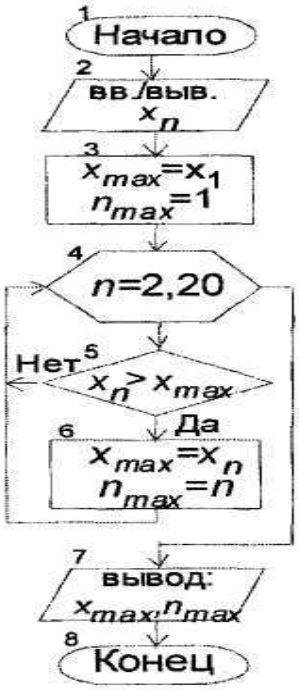


Рис. 8

Пример.

Разработаем алгоритм по нахождению наибольшего элемента массива x_i (x_1, x_2 ,...,

x_{20}) и его порядкового номера.

Здесь нет необходимости вычислять сравниваемые значения, так как они уже имеются в массиве x_i . Поэтому в качестве начальных значений x_{max} и номера n_{max} примем $x_{max} = x_1$ и $n_{max} = 1$ и сравнение будем производить по ц иклу, начиная со второго элемента массива x_i.

Схема алгоритма решения данного примера представлена на рис. 8.

Литература

1. Вальвачев, А.Н., Крисевич, В.С. Программирование на языке Паскаль для персональных ЭВМ ЕС. – Мн.: Выш. шк., 1989. – 223 с.
2. Паскаль для персональных компьютеров / Ю.С. Бородич [и др.]. – Мн.: Выш. шк., 1991. – 365 с.

3. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – М.: Госстандарт, 1990. – 28 с.

4. Офицеров, Д.В., Старых, В.А. Программирование в интегрированной среде Турбо-Паскаль. – Мн.: Беларусь, 1992. – 240 с.

5. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем.

6. Условные обозначения и правила выполнения. – М.: Госстандарт,1990.-28с.

МОДУЛЬ М2 – «БАЗОВЫЕ ЭЛЕМЕНТЫ АЛГОРИТМИЧЕСКОГО ЯЗЫКА ПАСКАЛЬ »

Лабораторная работа № 2

Запись чисел и переменных на языке Паскаль

***Цель работы*:** *Приобретение практических навыков записи на языке Паскаль чисел и переменных.*

Постановка задачи

Записать на Паскале по варианту условия, определяемому номером бригады, следующие данные (табл. 2.1 и табл. 2.2):

Таблица 2.1

Константы

№ п/п	Варианты							
	1	2	3	4	5	6	7	8
1	12,3	13,6	21,7	8,5	9,8	-6,7	7,34	8,19
2	-0,95	0,75	-0,85	-0,53	-0,79	0,24	-0,94	-0,37
3	5	6	3	4	5	6	7	8
4	4,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0
5	$1{,}3\cdot10^{5}$	$1{,}1\cdot10^{3}$	$1{,}4\cdot10^{2}$	$1{,}4\cdot10^{3}$	$1{,}7\cdot10^{5}$	$1{,}1\cdot10^{3}$	$1{,}5\cdot10^{4}$	$1{,}2\cdot10^{5}$
6	$-1{,}7\cdot10^{4}$	$-1{,}5\cdot10^{4}$	$-1{,}8\cdot10^{3}$	$-1{,}5\cdot10^{2}$	$-1{,}8\cdot10^{4}$	$-1{,}2\cdot10^{4}$	$-1{,}7\cdot10^{5}$	$-1{,}9\cdot10^{3}$
7	$-2{,}2\cdot10^{-3}$	$-0{,}7\cdot10^{-5}$	$-1{,}6\cdot10^{-4}$	$-1{,}7\cdot10^{-4}$	$-1{,}9\cdot10^{-3}$	$-1{,}3\cdot10^{-5}$	$-1{,}8\cdot10^{-6}$	$-2{,}7\cdot10^{-2}$

Таблица 2.2

Переменные

№ п/п	Варианты							
	1	2	3	4	5	6	7	8
1	α	ω	β	φ	π	ω	Δ	ρ
2	β_{12}	α_{13}	γ_{15}	β_{01}	φ_{05}	ρ_{06}	τ_{07}	ω_{06}
3	q_{zt}	r_s	v_r	q_z	p_v	r_z	q_ω	t_z
4	nos_1	st_{02}	les_3	bar_4	rab_5	vod_6	var_7	nok_8
5	$z_{i\,(i=\overline{1,n})}$	$v_{i\,(i=\overline{1,n})}$	$\omega_{k\,(k=\overline{1,n})}$	$\tau_{i\,(i=\overline{1,n})}$	$\sigma_{i\,(i=\overline{1,n})}$	$\eta_{k\,(k=\overline{1,n})}$	$\mu_{k\,(k=\overline{1,n})}$	$\delta_{k\,(k=\overline{1,n})}$
6	$\omega_{ij\,(i=\overline{1,n})\,(j=\overline{1,m})}$	$z_{ij\,(i=\overline{1,n})\,(j=\overline{1,m})}$	$\varepsilon_{ik\,(i=\overline{1,n})\,(k=\overline{1,m})}$	$\delta_{ij\,(i=\overline{1,n})\,(j=\overline{1,m})}$	$\psi_{ij\,(i=\overline{1,n})\,(j=\overline{1,m})}$	$\theta_{ik\,(i=\overline{1,n})\,(k=\overline{1,m})}$	$\lambda_{ik\,(i=\overline{1,n})\,(k=\overline{1,m})}$	$\beta_{ik\,(i=\overline{1,n})\,(k=\overline{1,m})}$
7	astra (симв. пер.)	bnty (симв. пер.)	bgpa (симв. пер.)	bpi (симв. пер.)	Bar (симв. пер.)	tost (симв. пер.)	fitr (симв. пер.)	post (симв. пер.)

Константы, переменные и их запись на языке Паскаль удобнее представить в виде следующих таблиц (табл. 2.3):

Таблица 2.3

Константы

№ п/п	Обычная запись	Паскаль	Тип
1	17,5	17.5	Real
…			

Переменные

№	Обыч-	Пас-
1	a	Alf,

Содержание лабораторной работы

Лабораторная работа включает:

1. Ознакомление с теоретическими сведениями (см. Приложение 2).
2. Оформление в отчете по лабораторной работе ответов на вопросы

согласно варианту задания.

Контрольные вопросы

1. Данные каких типов на языке Паскаль Вам известны?
2. В чем отличие понятий «константа» и «переменная»?
3. Какие формы записи констант на Паскале Вам известны?
4. Что такое идентификаторы? Каковы правила их записи на Паскале?

Содержание отчета

Отчет по выполненной работе должен содержать следующие сведения:

1. Номер и наименование лабораторной работы.
2. Цель работы.
3. Постановку задачи.
4. Ответы на вопросы задания.
5. Выводы по работе.

Лабораторная работа № 3

ЗАПИСЬ МАТЕМАТИЧЕСКИХ ВЫРАЖЕНИЙ НА ЯЗЫКЕ ПАСКАЛЬ

***Цель работы*:** *Приобретение практических навыков записи на языке Паскаль произвольных математических выражений.*

Постановка задачи

Записать на Паскале по варианту условия, определяемому номером бригады, следующие выражения (табл. 3.1):

Таблица 3.1

Варианты заданий

№ вариантов	№ п/п	Выражения
1	2	3
1	1	$y_1 = \frac{3{,}3x_1^2 + 25e_1^z}{\left\|x_1^7\right\| \cdot \sqrt{a_1 + z_1}}$
	2	$z_1 = \frac{a_1 + b_1}{e_1^x + \sin(x_1^2)} + \ln^2 \cdot x_1^2$
	3	$y_1 = a_1 + \frac{b_1 \cdot x_1}{e^{\alpha_1 \beta_1}}$
	4	$\gamma_1 = 3\sqrt{\cos^2(\alpha_1 + \rho_1) + \arcsin \beta_1}$
	5	$q_1 = \frac{1}{2\pi}\sqrt{\operatorname{arctg}\frac{\alpha_1}{\beta_1}}$
2	1	$y_2 = \frac{3{,}8y_1^2 + 65e^{x_1^2}}{\left\|y_1^7\right\| \cdot \sqrt{a_1 + z_1}}$
	2	$z_2 = \frac{c_1 + a_1}{e_2^y + \sin(x_2^2)} + \ln^2 \cdot x_2^3$
	3	$y_2 = a_2 + \frac{d_2 \cdot w_2}{e^{\alpha_2 \beta_2}}$

Продолжение табл. 3.1

1	2	3
	4	$w_2 = 3\sqrt{\cos^2(\alpha_2 + \rho_2) + \arccos \varepsilon_2}$
	5	$\gamma_2 = \frac{1}{2\pi}\sqrt[5]{\sin^3 \omega_2^3}$
3	1	$y_3 = \frac{7{,}3x_3^2 + 29x_3}{\lvert x_3^5 \rvert \cdot \sqrt{a_3 + z_3}}$
	2	$z_3 = \frac{a_3 + b_3}{e^{\beta} + \sin(x_3^2)} + \ln^2 \cdot x_3^2$
	3	$y_3 = a_3 + \frac{b_3 \cdot x_3}{e^{\alpha_3 \beta_3}}$
	4	$\gamma_3 = 3\sqrt{\cos^2(\alpha_3 + \rho_3) + \operatorname{arctg} s_3}$
	5	$t_3 = \arcsin\varphi_3 + \sqrt{\lvert \gamma_3^5 \rvert}$
4	1	$y_4 = \frac{3{,}8\, y_4^2 + 65e^{x_4^2}}{\lvert y_4^7 \rvert \cdot \sqrt{\beta_4 + z_4}}$
	2	$z_4 = \frac{c_4 + a_4}{e_4^y + \sin(x_4^2)} + \ln^2 \cdot x_4^2$
	3	$y_4 = a_4 + \frac{d_4 \cdot w_4}{e^{\alpha_4 \beta_4}}$
	4	$w_4 = 3\sqrt{\operatorname{tg}^2(\alpha_4 + \rho_4) + \arccos(\varphi_2)}$
	5	$\rho_4 = \arcsin \frac{\alpha_4}{\beta_4 \sqrt{\lvert \sigma_4 \rvert}}$

Продолжение табл. 3.1

1	2	3
5	1	$y_5 = \frac{3{,}8\,y_5^2 + 65e^{x_5^2}}{\lvert\varepsilon_5^2\rvert \cdot \sqrt{\alpha_5 + \beta_5}}$
	2	$z_5 = \frac{c_5 + a_5}{e_5^y + \sin(x_5^2)} + \ln^2 \cdot x_5^2$
	3	$y_5 = a_5 + \frac{d_5 \cdot w_5}{e^{\alpha_5 \beta_5}}$
	4	$w_5 = 3\sqrt{\cos^2(\alpha_5 + \rho_5) + \operatorname{arctg} \varphi_5}$
	5	$t_5 = \arcsin \frac{x_5}{\beta_5} + e^z$
6	1	$y_6 = \frac{3{,}8\,f_6^2 + 65e^{x_6^2}}{\lvert z_6^5\rvert \cdot \sqrt{\alpha + \varepsilon}}$
	2	$z_6 = \frac{c_6 + a_6}{e_6^y + \sin(x_6^2)} + \ln^2 \cdot x_6^3$
	3	$y_6 = a_6 + \frac{d_6 \cdot w_6}{e^{\alpha_6 \beta_6}}$
	4	$w_6 = 3\sqrt{\cos^2(\alpha_6 + \rho_6) + \arcsin \alpha_6}$
	5	$r_6 = \arcsin \frac{\gamma_6}{\beta_6} + e^{\sqrt{\ln(x)\lvert}}$
7	1	$y_7 = \frac{13{,}8\,w_7^2 + 65x_7^3}{\lvert\varepsilon_7^5\rvert \cdot \sqrt{\alpha + \lambda}}$

Окончание табл. 3.1

1	2	3
	2	$z_7 = \frac{c_7 + a_7}{e_7^y + \sin(x_7^2)} + \ln^2 \cdot x_7$
	3	$y_7 = a_7 + \frac{d_7 \cdot \sin(w_2)}{e^{\alpha_7 \beta_7}}$
	4	$w_7 = 3\sqrt{\cos^2(\alpha_7 + \rho_7) + \operatorname{arctg} \omega_7}$
	5	$q_7 = \arccos \frac{\gamma_7}{\beta_7} + e^{2h}$
8	1	$y_8 = \frac{9{,}7\,y_2^2 + 2{,}6\,x_8^2}{\lvert\delta_8^7\rvert \cdot \sqrt{a_8 + z_8}}$
	2	$z_8 = \frac{c_1 + a_1}{e_8^y + \sin(x_8^2)} + \ln^2 \cdot x_8^3$
	3	$y_8 = \operatorname{tg}(a_8) + \frac{z_8 \cdot w_8}{e^{\alpha_8 \beta_8}}$
	4	$w_8 = 3\sqrt{\sin^2(\alpha_8 + \rho_8) + \operatorname{arctg}(\varphi_8)}$
	5	$q_8 = \arccos \frac{x_8}{\beta_8} + e^{\sqrt{t}}$

Содержание лабораторной работы

Лабораторная работа включает:

1. Ознакомление с теоретическими сведениями (см. Приложение 2).
2. Оформление в отчете по лабораторной работе ответов на вопросы согласно варианту задания.

Контрольные вопросы

1. Что представляет собой математическое выражение на языке Паскаль?
2. Что такое операнд? Какими знаками связаны между собой операнды в выражениях?
3. Какие математические операции на Паскале Вам известны?
4. Что такое стандартные функции Паскаля? Каковы правила их записи?
5. Каковы правила записи выражений на Паскале?
6. Что такое приоритет выполнения операций в выражениях Паскаля?

Содержание отчета

Отчет по выполненной работе должен содержать следующие сведения:

6. Номер и наименование лабораторной работы.
7. Цель работы.
8. Постановку задачи.
9. Ответы на вопросы задания.
10. Выводы по работе.

Приложение 2 (к модулю М2)

Теоретические сведения к лабораторным работам № 2- «Запись чисел и переменных на языке Паскаль» и № 3 - «Запись математических выражений на языке Паскаль»

1. *Вводные сведения о Паскале и системе программирования Турбо-Паскаль*

Алгоритмический язык Паскаль является одним из популярных и широко распространённых языков программирования инженерных задач. Впервые его описание было опубликовано в 1971г. создателем этого языка Никлаусом Виртом - профессором Цюрихского технологического института (Швейцария). Название язык получил в честь французского математика и философа Блеза Паскаля (1623-1662). Первый транслятор с языка Паскаль был разработан в 1973г. Этот язык лёгок в изучении и удобен для программирования (набор его операторов относительно мал), является наиболее совершенным по сравнению с Бейсиком, Фортраном и др. языками и в 1982г. утверждён в качестве международного стандарта.

В начале 80-х годов появилась первая версия языка Паскаль как составная часть системы программирования для ПК. В настоящее время одной из наиболее популярных систем программирования для ПК является Турбо-Паскаль (ТП), представляющая собой интегрированную среду и включающая: экранный редактор, компилятор, редактор связей и отладчик. Интегрированность среды проявляется не только в единой концепции построения её составляющих частей, но и в связи их друг с другом. Так, при возникновении ошибки трансляции система ТП автоматически переходит в режим экранного редактирования и ставит курсор в точку возникновения ошибки. Аналогичные действия выполняются отладчиком при возникновении ошибки во время выполнения программы.

Система программирования ТП создана американской фирмой Borland International. В первой версии этой системы был объединён очень быстрый компилятор с редактором текста. В 1985г. на рынке ПК появилась версия 3.0 системы программирования ТП с компилятором стандартного Паскаля, которая благодаря простоте её использования получила широкое применение. В пакете ТП версии 4.0 было устранено большинство подвергавшихся критике огра-

ничений компилятора и была повышена производительность системы программирования, что дало возможность разработки в этой системе крупных программных продуктов. В ТП версии 5.0 в среду программирования был встроен интегрированный отладчик, позволяющий повысить производительность труда программистов. В версии ТП 5.5 улучшены технические характеристики: наряду с внутренними улучшениями реализована концепция объектно-ориентированного программирования. В ТП версии 6.0 чисто теоретическая концепция объектно-ориентировочного программирования реализована практически с полным набором объектов, которые могут использоваться для решения прикладных задач. Кроме того, реализация системы меню приведена в соответствие со стандартом SAA (Turbo Vision); реализован текстовый редактор, встроенный в IDE (интегрированную инструментальную оболочку). В 1992 г. фирмой Borland International разработана была очередная версия 7.0 системы программирования ТП, в которой унаследованы преимущества предыдущей версии и произведены некоторые изменения и улучшения: появилась возможность выделять определённым цветом различные элементы исходного текста программы (ключевые слова, идентификаторы, числа и т.д.), улучшен компилятор (коды программ стали более эффективными), улучшен интерфейс пользователя и др.

2. Базовые элементы языка Паскаль

Алфавит и словарь. При записи алгоритма решаемой задачи на языке Паскаль используется конечный набор знаков (символов), образующих ***алфавит*** этого языка.

Он состоит из *прописных и строчных* ***букв*** *латинского алфавита*

ABCDEFGHIJKLMNOPQRSTUVWXYZ

a b c d e f g h i j k l m n o p q r s t u v w x y z,

знака подчёркивания _

цифр **0 1 2 3 4 5 6 7 8 9**

и *специальных символов*

+ плюс	{} фигурные скобки

-	минус	.	точка
*	звёздочка	,	запятая
/	дробная черта	:	двоеточие
=	равно	;	точка с запятой
>	больше	'	апостроф
<	меньше	#	номер
[]	квадратные скобки	$	номер денежной единицы
()	круглые скобки	^	тильда
_	пробел		

Комбинации специальных символов образуют *составные символы:*

:=	присвоить	<=	меньше или равно
<>	не равно	>=	больше или равно
..	диапазон значений	**(..)**	альтернатива квадратных скобок

(* *) альтернатива фигурных скобок.

В комментариях к программе и в символьных константах могут использоваться любые другие знаки, например буквы русского алфавита. Комментарий представляется следующей конструкцией:

{Любой текст}

и может быть помещен в любом месте программы.

Неделимые последовательности знаков алфавита образуют *слова,* которые несут определённый смысл в программе и отделяются друг от друга *разделителями* (пробелом, символом конца строки, комментарием). Слова делятся на ключевые (зарезервированные) слова, стандартные идентификаторы и идентификаторы пользователя.

Ключевые слова имеют фиксированное написание и определённый смысл.

Идентификатором *называется последовательность букв и цифр, начинающаяся с буквы или знака подчёркивания.* Идентификаторы применяются для обозначения имён переменных, констант, функций и процедур (подпрограмм).

Стандартные идентификаторы - это обозначения заранее определённых разработчиками языка Паскаль типов данных, констант, процедур и функций. Например, стандартный идентификатор **sin(x)** вызывает функцию по вычислению синуса заданного угла **x.**

Идентификаторы пользователя применяются для обозначения меток, констант, переменных, процедур и функций, которые задаются самим пользователем. Например, дату удобнее обозначать идентификатором **Data,** чем просто буквой D или каким-либо другим символом. Существуют *общие правила написания идентификаторов пользователя:*

- идентификатор должен начинаться только с буквы или знака подчёркивания, исключение, составляют метки, которые могут начинаться также и с цифры;

- для обозначения идентификаторов используются только буквы, цифры и знак подчёркивания;

- между двумя идентификаторами должен быть, по крайней мере, один пробел;

- идентификаторы могут быть любой длины (до 126 символов), но сравнение их между собой производится по первым 63 символам;

- при написании идентификаторов можно использовать как прописные, так и строчные буквы, так как компилятор не делает различий между ними. Поэтому на практике для более простого чтения и понимания целесообразно, например, написать ***NomerOtdela*** (вместо ***nomerotdela),*** выделив прописными буквами каждую из двух смысловых частей этого идентификатора.

Константы и переменные. При решении любой задачи по программе используются конкретные данные, над которыми выполняются определённые действия для получения результата решения. В программе каждый элемент данных является или константой, или переменной.

Константами *называются элементы данных, значения которых заданы перед решением задачи и которые в процессе выполнения программы не изменяются. Формат их описания:*

Const идентификатор = значение константы;

Имеется ряд стандартных констант, к значениям которых можно обращаться без предварительного описания, например:

Maxint = 32767;

True = истинно;

False = ложно;

Pi =3,14159…

Переменная *- это именованный объект, который в процессе выполнения программы может принимать различные значения.*

Переменные имеют следующий формат описания:

Var идентификаторы: *тип;*

В Турбо-Паскале используются три вида констант:

*числовые (целые или вещественные);

*логические (или булевские);

*символьные и строковые.

Целые константы *— это положительные или отрицательные целые числа (без десятичной точки).* Знак «+» в положительных числах можно опускать.

Турбо-Паскаль позволяет использовать также шестнадцатеричные целые значения. При использовании шестнадцатеричной константы перед ее значением указывается знак доллара **$**. Например, **$27** определяет десятичное число 39.

Вещественные константы могут быть представлены в двух формах: с фиксированной и плавающей точкой. *Константы с фиксированной точкой* - это числа, содержащие точку, разделяющую целую и дробную части. *Константы с плавающей точкой* - это числа, представленные с десятичным порядком **mEp** (без пробелов), где **m** - мантисса (как целые, так и вещественные числа

с фиксированной точкой); **E** - признак записи числа с десятичным порядком; **p** - порядок числа (только целые числа).

Пример.

Значение константы	Целая константа	Константа с фиксированной точкой	Константа с плавающей точкой
-257	-257	-257.0	-2.57E2
16,4	--	16.4	1.64E1
0,0032	--	0.0032	0.32E-2

Константы с фиксированной точкой обязательно должны содержать как целую, так и дробную часть: 2.0; 0.5; -13.0.

Логические константы *- это константы, которые принимают только два значения: True (истинно) или False (ложно).*

Символьные константы *- это какой-либо один символ, заключенный в апострофы: 'A', '1'.*

Строковые константы *- это ряд символов, заключенных в апострофы: '+9CL', 'A и B'.*

Кроме констант и переменных существуют ещё так называемые ***типизированные константы,*** которые являются промежуточными данными между константами и переменными. Слово *константа* означает, что типизированная константа описывается в разделе **Const,** а слово *типизированная* - что должен указываться ***тип,*** как у переменных. Формат описания их следующий;

Const идентификатор: ***тип*** = значение;

Типы данных. Константы, переменные, функции, выражения, которые используются в программе, относятся к определённому типу. ***Тип*** - это множество значений элементов программы и совокупность операций над ними. Например, значения 1 и 5 относятся к целочисленному типу, над ними можно выполнять различные арифметические операции.

В Паскале типы данных делятся на *скалярные* (стандартные и пользовательские) и *структурированные* (содержащие различные комбинации скалярных типов).

К стандартным скалярным типам данных относятся: целочисленные, байтовые, вещественные, символьные и булевские типы.

Целочисленный тип данных включает все целые числа в диапазоне от -32768 до +32767. Для размещения в памяти значения переменной целочисленного типа требуется 2 байта. Формат описания целочисленных переменных следующий:

Var идентификаторы: integer;

Байтовый тип данных аналогичен целочисленному, но охватывает меньший диапазон значений: от нуля до 255. Переменная байтового типа занимает в памяти 1 байт и имеет следующий формат описания:

Var идентификаторы: byte;

Вещественный тип данных включает положительные и отрицательные числа от

$1 * 10^{-38}$ до $1 * 10^{+38}$, при этом мантисса может содержать до 11 значащих цифр. Данные этого типа могут записываться с фиксированной (целая часть числа от дробной отделяется точкой) и плавающей точкой в виде **mE±p,** где **m** - мантисса, представляющая собой целое или дробное число с десятичной точкой; **E** означает «десять в степени»; **p** - порядок в виде двухзначного целого числа.

Переменная вещественного типа в памяти занимает 6 байт и имеет следующий формат описания:

Var идентификаторы: rea1;

Переменная *булевского типа* в памяти занимает 1 байт, может принимать одно из двух значений (констант): **True** (истина) или **False** (ложь) и имеет следующий формат описания:

Var идентификаторы: *boolean;*

Константы и переменные ***символьного (литерного) типа*** принимают одно из значений кодовой таблицы ПК. Переменная этого типа в памяти занимает 1 байт. Конкретные значения переменных и констант символьного типа записываются в апострофах. Например, 'A' представляет собой букву A,

';' - точку с запятой. Формат описания переменной символьного типа следующий:

Var идентификаторы: char;

К скалярным пользовательским типам данных относятся перечисляемый и интервальный типы. Данные этих типов в памяти занимают по 1 байту.

Перечисляемый тип данных задаётся перечислением всех значений в круглых скобках через запятую. Форматы описания их следующие:

1. **Type** имя типа = (значение 1,... , значение n);

   Var идентификаторы: имя типа;

или

2. **Var** идентификаторы: (значение 1, ... , значение n**);**

Пример:

```
Type Gaz = (C,O,N,F);
     Metall = (Fe,Co,Na,Cu,Zn);
Var Gl,G2:Gaz;
Metl,Met2:Metall;
Ses : (Winter, Spring, Summer, Autumn);
```

В этом примере явно описаны 2 типа данных пользователя - Gaz и Metall (перечислены некоторые газы и металлы периодической таблицы Д.И. Менделеева). Переменные Gl, G2 и Metl, Met2 могут принимать только одно из перечисленных значений. Третий тип перечисления (Ses) анонимный, так как не имеет имени типа. Он задаётся перечислением значений переменных в разделе описаний **Var** (т.е. записан по второму формату).

Интервальный тип данных задаётся двумя граничными константами диапазона значений для данной переменной. Обе константы должны быть одного из стандартных типов, кроме вещественного, при этом значение первой константы должно быть меньше значения второй. Формат описания:

Type имя типа = константа 1 .. константа 2;

Var идентификаторы: имя типа;

Пример: **Type** Dni=l ..31;

Var RabDni, VolnDni: Dni;

В этом примере переменные RabDni и VolnDni имеют интервальный тип Dni и могут принимать любые значения из диапазона 1 ..31. Выход из диапазона вызывает программное прерывание.

Границы диапазона можно задавать не значениями констант, а их именами:

Const Min= I; Max=31;

Type Dni=Min .. Max;

Var RabDni, VolnDni: Dni;

Структурированные типы данных (строки, массивы, записи, множества, файлы и указатели) представляют собой упорядоченные определённым образом совокупности скалярных переменных и характеризуются типом своих элементов. Приведём лишь краткую характеристику этих данных (данные типа массивов, которые часто используются в вычислительных алгоритмах инженерных задач, более подробно рассмотрены в лабораторной работе № 9).

Строка - *это последовательность символов, заключённая в апострофы.*

Массив - *это данные, состоящие из фиксированного количества элементов, имеющих один и тот же тип и объединённых под общим именем.*

Множество - *это набор выбранных по какому-то признаку (или группе признаков) объектов, которые можно рассматривать как единое целое.*

Запись - *это данные, состоящие из фиксированного числа элементов разного типа.*

Файл - *это данные, состоящие из последовательности элементов одного типа и одной длины.* Чаще всего элементами файла являются записи.

Указатель - *это структурированный тип данных, состоящий из неограниченного множества указывающих на однотипные элементы значений.* Используется при работе с динамическими структурами данных.

Стандартные арифметические функции. Они реализуют наиболее часто встречающиеся математические действия и операции. Приведём их описание и примеры использования, обозначив через **x** целочисленные и вещественные типы.

Abs(x) - вычисление абсолютной величины выражения **x**. Тип результата совпадает с типом параметра **x**.

Abs(4-6)=2

ArcTan(x) - вычисление угла, тангенс которого равен **x**; значение угла представляется в радианах в диапазоне от -Pi/2 до Pi/2. Результат имеет вещественный тип.

ArcTan(l)=Pi/4

Sin(x), Cos (x) - вычисление синуса или косинуса выражения **x**; **x** - в радианах; результат имеет вещественный тип.

Cos(60*Pi/180)=0.5

Exp(x) - вычисление экспоненты **x**, т.е. значение **e** в степени **x** (**e** - основание натурального логарифма, равное 2,718282). Результат имеет вещественный тип.

Exp(1)=2.718282

Frac(x) - вычисление дробной части выражения **x**; результат имеет вещественный тип.

Frac(0.25*11)=0.75

Int(x) - вычисление целой части **x**.

Int (123.448)=123

Int(-345.555)=-345

Sqr(x) - возведение в квадрат значения **x**. Тип результата совпадает с типом аргумента **x**.

Sqr(5)=25

Sqrt(x) - вычисление квадратного корня из **x**. Тип результата вещественный.

Sqrt(25)=5.0

Ln(x) - вычисление натурального логарифма по основанию **e**. Результат имеет вещественный тип.

$$Ln(3)=1.0986$$

Для вычисления логарифма с основанием **a** используется соотношение:

$$\log_a(x)=\ln(x)/\ln(a)$$

Для вычисления других тригонометрических функций следует использовать известные соотношения:

$Tan(x)=sin(x)/cos(x)$ $Ctg(x)=cos(x)/sin(x)$

$Csc(x)=l/sin(x)$ $Sc(x)=l/cos(x)$

$ArcSin(x)=ArcTan(x/(l-x^2))^{1/2}$

$ArcCos(x)=Pi/2-ArcSin(x)$

$ArcCtg(x)=Pi/2-ArcTan(x)$

В языке Паскаль нет операции возведения в степень. Поэтому эту операцию заменяют другими с применением стандартных функций Exp и Ln:

$$X^a=Exp(a*Ln(x))$$

Вычисление выражения $(-x)^n$, если n - целочисленная константа, организуют по циклу путём умножения (-x) само на себя n раз.

Выражения, операнды и операции. ***Выражение*** задаёт порядок выполнения действий над элементами данных и состоит из *операндов* (констант, переменных, обращений к функциям), *знаков операций* и *круглых скобок.* ***Операции*** определяют действия, которые необходимо выполнить над операндами. В простейшем случае выражение может состоять из одного операнда. Круглые скобки ставятся так же, как в обычных арифметических выражениях для управления порядком выполнения операций. Их использование также вполне приемлемо и полезно для более чёткого и понятного визуального определения порядка вычислений.

Операции подразделяются на арифметические, отношения, логические, строковые и др. Выражения соответственно бывают арифметические, отношения, логические, строковые и др. в зависимости от того, какого типа операнды и операции в них используются.

Операции могут быть *унарными* (относятся к одному операнду и записывается перед ним) и *бинарными* (выражают действия над двумя операндами и записываются между ними). Например, в выражении -A символ "-" является унарной операцией, а в выражении A-B - бинарной.

Под ***арифметическим выражением*** понимают совокупность констант, переменных и функций, объединённых знаками арифметических операций и круглыми скобками. Результатом вычисления арифметического выражения всегда является число. Арифметические операции языка Паскаль представлены в табл. 1.

Таблица 1

Арифметические операции.

Знак	Опрерация	Выражение
+	Сложение	A+B
-	Вычитание	A - B
*	Умножение	A*B
/	Деление	A/B
Div	Целочисленное деление	A div B
Mod	Деление с остатком	A mod B
And	Арифметическое И	A and B
Shr	Целочисленный сдвиг вправо	A shr B
Shl	Целочисленный сдвиг влево	A shl B
Or	Арифметическое ИЛИ	A or B
Xor	Исключающее ИЛИ	A xor B
-	Изменение знака	- A

Операции сложения (+), вычитания (-), умножения (*) и деления (/) выполняется так же, как и в обычных арифметических выражениях.

Целочисленное деление (div) отличается от обычной операции деления тем, что результатом является целая часть частного, при этом дробная часть от-

брасывается. Результат целочисленного деления всегда равен нулю, если делимое меньше делителя.

Примеры.	Выражение	Результат
	10 div 3	3
	2 div 3	0

Деление по модулю (mod) восстанавливает остаток, полученный при выполнении целочисленного деления.

Примеры.	Выражение	Результат
	10 mod 3	1
	2 mod 3	2

В выражениях для *арифметических операций* **and, shr, shl, or, xor** операнды записываются в десятичной форме, а при выполнении операций предварительно переводятся в двоичную систему счисления. Результаты поразрядного вычисления затем представляются в десятичной форме.

Примеры.	Выражение	Результат
	12 and 22	4
	2 shl 7	256
	160 shr 2	40
	12 or 22	30
	12 xor 22	26

Операция (унарная) изменения знака восстанавливает значение операнда с противоположным знаком.

Примеры.	Выражение	Результат
	-(-256)	256
	-(+39)	-39

При написании арифметических выражений рекомендуется соблюдать следующие ограничения и правила:

1. Запрещено последовательное написание двух операций, поэтому запись A/-B запрещается, A/(-B) разрешается.

2. Приоритет операций в порядке убывания следующий:

1) *, /, div, mod, and, shl; shr;

2) +, -, or, xor.

3. Часть выражения, заключённая в круглые скобки, выполняется в первую очередь.

4. Операции одинакового приоритета в выражении выполняются последовательно слева направо.

Выражение отношения состоит из двух или более арифметических выражений, соединённых операциями отношения. Оно определяет истинность или ложность результата, который имеет логический (булевский) тип и принимает одно из двух значений: **True** (истина) или **False** (ложь). Операции отношения на Паскале записывается так: = (равно), <> (не равно), > (больше), < (меньше), >= (больше или равно), <= (меньше или равно), **in** (принадлежность). Все операции являются бинарными.

При объединении в одном выражении арифметических операций и операций отношения первыми всегда выполняются арифметические операции. *Сравниваемые данные должны быть одинакового типа.*

Логическое выражение образуется из операндов логического типа и логических операций: **not** (логическое отрицание), **and** (логическое умножение, И), **or** (логическое сложение, ИЛИ), **xor** (исключающее ИЛИ). При этом операндами могут быть: логические константы, логические переменные, выражения отношения.

Старшинство логических операций в порядке убывания следующее: 1) **not,** 2) **and, 3)or, xor.** Приоритет логических операций выше операций отношения (в других алгоритмических языках наоборот).

При вычислении логического выражения, содержащего различные операции (арифметические, логические и отношения), выполнение каждой операции осуществляется с учётом её приоритета (табл.2).

Таблица 2

Порядок выполнения операций.

Операция	Приоритет	Вид операции
Not	Первый (высший)	Унарная операция
, /, div, mod, and, shl, shr	Второй	Операции типа умножения
+, -, or, xor	Третий	Операции типа сложения
=, <>, <, >, <=, >=, in	Четвёртый (низший)	Операции отношения

Для выполнения операций не по старшинству применяются круглые скобки. Например, в выражении **(A<C) and (B=D)** операции отношения будут выполняться раньше, чем логическая операция **and**.

Результатом вычисления логического выражения является константа логического типа.

Выражения 1-5 (см. табл. 3.1) программируются с помощью операторов присваивания. Общий вид оператора присваивания следующий:

Идентификатор переменной := выражение;

Здесь *идентификатор переменной* – имя переменной, текущее значение которой заменяется новым значением, определяемым данным *выражением*.

Пример.

```
Y := Sqrt (x)+1;
b := M and N;
```

В операторе присваивания *идентификатор переменной* и *выражение* должны иметь один и тот же тип, кроме одного исключения: идентификатору переменной типа **real** разрешается присваивать выражение типа **integer**.

Литература

1. Вальвачев, А.Н., Крисевич, В.С. Программирование на языке Паскаль для персональных ЭВМ ЕС. – Мн.: Выш. шк., 1989. – 223 с.
2. Паскаль для персональных компьютеров / Ю.С. Бородич [и др.]. – Мн.: Выш. шк., 1991. – 365 с.

3. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – М.: Госстандарт, 1990. – 28 с.
4. Офицеров, Д.В., Старых, В.А. Программирование в интегрированной среде Турбо-Паскаль. – Мн.: Беларусь, 1992. – 240 с.

МОДУЛЬ М3 – «ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ АЛГОРИТМОВ»

Лабораторная работа № 4

ВВОД-ВЫВОД ДАННЫХ НА ЯЗЫКЕ ПАСКАЛЬ

Цель работы*: *Приобретение практических навыков организации ввода-вывода данных на языке Паскаль.

Постановка задачи

Осуществить ввод-вывод данных на Паскале по варианту условия, определяемому номером бригады ,(табл. 4.1). При этом предусмотреть для данных колонки Ввода-вывода использование операторов **Read-Write,** а для данных колонки Вывода - использование операторов **Присваивания** и **Writeln.**

Таблица 4.1

Варианты заданий

№ вариантов	Условия	
	Ввода-вывода	Вывода
1	2	3
1	$\alpha_1 = 1,5$; $t_1 = 1,3 \cdot 10^{-5}$; $r_1 = -3$; $\beta_1 = -36,2$	$z_1 = 16,3$; $x_1 = -2,1 \cdot 10^{-3}$; $y_1 = 5$; $\omega_1 = 0,75$
2	$\varepsilon_2 = 7 \cdot 10^{-4}$; $z_2 = 1,62$; $c_2 = 6$; $d_2 = -1,362$	$\omega_2 = 1,3 \cdot 10^{-2}$; $f_2 = -22,1$; $g_2 = 10$; $h_2 = 0,935$
3	$\beta_3 = 9$; $\omega_3 = 1,63 \cdot 10^{-7}$; $a_3 = -0,3$; $\alpha_3 = 12,62$	$\omega_3 = -3,63$; $p_3 = 3,1 \cdot 10^{4}$; $z_3 = 0,3$; $y_3 = 9$
4	$\delta_4 = 11,39$; $q_4 = 2,5 \cdot 10^{-3}$; $f_4 = 8$; $z_4 = -0,762$	$q_4 = 19$; $b_4 = 0,4 \cdot 10^{4}$; $c_4 = -465$; $n_4 = 195$

Окончание табл. 4.1

1	2	3
5	$\gamma_5 = 1{,}1 \cdot 10^{-5}$; $t_5 = 12{,}5$; $\omega_5 = 0{,}52$; $n_5 = 15$	$\delta_5 = 1{,}89$; $d_5 = 0{,}5 \cdot 10^{5}$; $f_5 = -79$; $m_5 = 137$
6	$\omega_6 = 13{,}8$; $h_6 = 6{,}2 \cdot 10^{-5}$; $k_6 = 2003$; $\alpha_6 = -0{,}75$	$t_6 = 19{,}8$; $\beta_6 = 7{,}9 \cdot 10^{-3}$; $j_6 = -13$; $m_6 = 15$
7	$\beta_7 = 6{,}79$; $z_7 = 5{,}1 \cdot 10^{-5}$; $x_7 = -0{,}863$; $l_7 = 12$	$p_7 = -8{,}91$; $\omega_7 = -27$; $\alpha_7 = 8{,}1 \cdot 10^{-6}$; $\gamma_7 = 0{,}21$
8	$\beta_8 = 11$; $\omega_8 = 1{,}33 \cdot 10^{-6}$; $a_8 = -0{,}5$; $\alpha_8 = 13{,}56$	$\omega_8 = -3{,}63$; $p_8 = 2{,}1 \cdot 10^{3}$; $z_8 = 0{,}73$; $y_8 = 19$

Содержание лабораторной работы

Лабораторная работа включает:

1. Ознакомление с теоретическими сведениями (см. Приложение 3).

2. Оформление отчета по лабораторной работе, который должен содержать следующие пункты:

1)номер и название работы;

2)цель работы;

3)постановку задачи;

4)схему алгоритма;

5)таблицу идентификаторов;

6)текст исходной Паскаль-программы.

Порядок выполнения работы

Последовательность выполнения работы следующая:

1. Набрать на клавиатуре текст Паскаль-программы .

2. Произвести компиляцию исходной программы.

3. В случае обнаружения ошибок отредактировать программу с ее последующей повторной компиляцией.

4. Запустить программу после сообщения об ее успешной компиляции.

5. Ввести исходные данные для получения окончательного результата.

6. Распечатать текст Паскаль-программы и результаты.

Контрольные вопросы

1.Что Вы понимаете под вводом данных на Паскале?

2.В какой форме осуществляется ввод данных на Паскале?

3.Что Вы понимаете под выводом данных на Паскале?

4.В каких форматах можно осуществить вывод данных на Паскале?

5.Какими операторами можно организовать ввод данных на Паскале?

6.Каковы правила записи операторов ввода-вывода данных на Паскале?

Содержание отчета

Отчет по выполненной работе (кроме протокола) должен содержать следующие сведения:

1. Экспериментальные результаты в виде распечатки текста отлаженной Паскаль-программы и результатов счета.
2. Выводы по работе.

Лабораторная работа № 5

Программирование линейных вычислительных процессов

Цель работы*: *Приобретение практических навыков составления Паскаль-программ решения задач линейных вычислительных процессов.

Постановка задачи

Разработать блок-схемы алгоритмов и составить Паскаль-программы решения задач по варианту условия, определяемому номером бригады (табл. 5.1 и 5.2).

Таблица 5.1

Варианты заданий

№ вариантов	Математические выражения	Исходные данные
1	2	3
1	$\alpha_1 = \frac{ax^2 + \sin^2 z}{\sqrt{1 + e^y}}$	a = 15,2; x = 0,89; z = 31,8; y = 1,25
2	$t_2 = \frac{\beta^2 + \sqrt{\lvert q \rvert}}{\cos^2 x + \beta \ln x}$	β = 0,85; q = 10,2; x = 2,675
3	$q_3 = \frac{\sin^2 (z + a)^3}{t\sqrt[3]{e^{a+2q}}}$	z = 0,764; a = 1,27; t = 12,5; q = 0,9
4	$z_4 = \frac{3x^2 - \sqrt{\cos(y^3)}}{\ln^2 (y + \gamma)}$	x = 2,61; y = 1,13; γ = 0,84
5	$\lambda_5 = \frac{4q\sqrt{\lvert x + \sin(z^3) \rvert}}{3\ln^2 (q + x)}$	q = 7,6; x = -0,78; z = 4,67
6	$\delta_6 = \frac{a\sqrt[3]{x + \ln^2 y}}{\lvert t^3 \rvert}$	a = 1,8; x = 0,729; y = 6,3; t = -1,5
7	$\varphi_7 = \frac{\rho^3 + e^{2\beta t}}{13{,}2\sqrt{\ln(\alpha + t)}}$	ρ = 0,875; α = 1,8; t = 7,9; β = 1,1
8	$\omega_8 = \frac{\lvert \alpha^3 \rvert + \sqrt[3]{\sin^2 z}}{\sqrt{x}\, e^{\alpha t}}$	α = 2,65; z = 1,7; x = 15,4; t = 0,76

Таблица 5.2

Варианты заданий

№ вариантов	Логические выражения	Арифметические выражения	Примечания
1	B1 = (a < 1) ^ (b > 7)	c = d or f	1. Значениями исходных данных задаться самостоятельно;
2	B2 = (a >= 10) v (c1 < 15)	c = d shl 2	
3	B3 = (s <= (t + 1)) ^ (g < 5)	c = d and f	
4	B4 = m v (f > (k + 1))	c = d xor k	2. Символ ^ оз-

5	B5 = (c > b1) ^ (f < k)	c = d shr 3	начает логическую операцию «И», а символ v – логическую операцию «ИЛИ»
6	B6 = (x < y) v (g > 2)	c = d and k	
7	B7 = (2a = x1) ^ (z1 < 8)	c = d xor k1	
8	B8 = ((k + 1) = c1) ^ x1	c = k1 and f	

Содержание лабораторной работы

Лабораторная работа включает:

1. Ознакомление с теоретическими сведениями (см. Приложение 3).

2. Оформление отчета по лабораторной работе, который должен содержать следующие пункты:

1) номер и название работы;
2) цель работы;
3) постановку задачи;
4) блок-схему алгоритма решения задачи;
5) таблицу идентификаторов;
6) текст исходной Паскаль-программы.

Порядок выполнения работы

Последовательность выполнения работы следующая:

1. Набрать на клавиатуре текст Паскаль-программы .

2. Произвести компиляцию исходной программы.

3. В случае обнаружения ошибок отредактировать программу с ее последующей повторной компиляцией.

4. Запустить программу после сообщения об ее успешной компиляции.

5. Ввести исходные данные для получения окончательного результата.

6. Распечатать текст Паскаль-программы и результаты.

Контрольные вопросы

1. Что Вы понимаете под термином «линейный вычислительный процесс»?
2. Как строится схема алгоритма линейного вычислительного процесса?
3. Каково назначение таблицы идентификаторов?
4. С чего начинается написание Паскаль-программы?
5. Чем заканчивается текст Паскаль-программы?
6. В каких форматах можно осуществить вывод данных на Паскале?
7. Что означает выражение «естественный порядок выполнения операторов»?

Содержание отчета

Отчет по выполненной работе (кроме протокола) должен содержать следующие сведения:

1. Экспериментальные результаты в виде распечатки текста отлаженной Паскаль-программы и результатов счета.
2. Выводы по работе.

Приложение 3 (к модулю М3)

Теоретические сведения к лабораторным работам:

№ 4- «ВВОД-ВЫВОД ДАННЫХ НА ЯЗЫКЕ ПАСКАЛЬ» и №5- «Программирование линейных вычислительных процессов»

1. Структура и общие правила написания программы на Паскале

Программа реализует алгоритм решения задачи и представляет собой последовательность действий над определёнными данными с помощью математических операций. При разработке программы следует руководствоваться *основными принципами структурного программирования:*

- не используйте сложных методов там, где можно обойтись простыми;
- без крайней необходимости не используйте оператор перехода **goto**;

- исключайте переходы извне внутрь рассматриваемой разветвляющейся структуры;

- циклические структуры задавайте в явном виде, избегая операторов **goto**;

- большие программы разбивайте на логически завершённые сегменты (процедуры и функции);

- выбирайте имена констант, переменных, процедур, функций по смыслу, с учётом их назначения.

Программа на языке Паскаль состоит из строк. Набор текста программы осуществляется с помощью встроенного редактора текстов системы программирования Турбо-Паскаль. Существуют различные схемы написания программ на Паскале, которые отличаются количеством отступов слева в каждой строке и различным использованием прописных букв. Строка может начинаться с любой колонки, т.е. величина отступа от левой границы для каждой строки устанавливается самим программистом с целью получения наиболее ясного текста программы. Количество операторов в строке произвольно. Один оператор может записываться на нескольких строках. Такое разбиение является условным из соображения удобства и чёткости, так как никаких знаков переноса в Паскале не используется.

Синтаксически программа состоит из необязательного заголовка и программного блока. Заголовок в общем случае состоит из ключевого слова Program и имени программы.

Программный блок может содержать в себе другие блоки. Блок, который не входит ни в какой другой блок, называется *глобальным*. Другие блоки, находящиеся в глобальном блоке, называются *локальными*. ***Глобальный блок*** - это основная программа, локальные блоки - это процедуры и функции. Отдельные элементы программы (типы, переменные, константы и др.) соответственно называются глобальными или локальными и областью действия их являются: блок, в котором они описаны, и все вложенные в него блоки. Блочная структура обеспечивает структуриза-

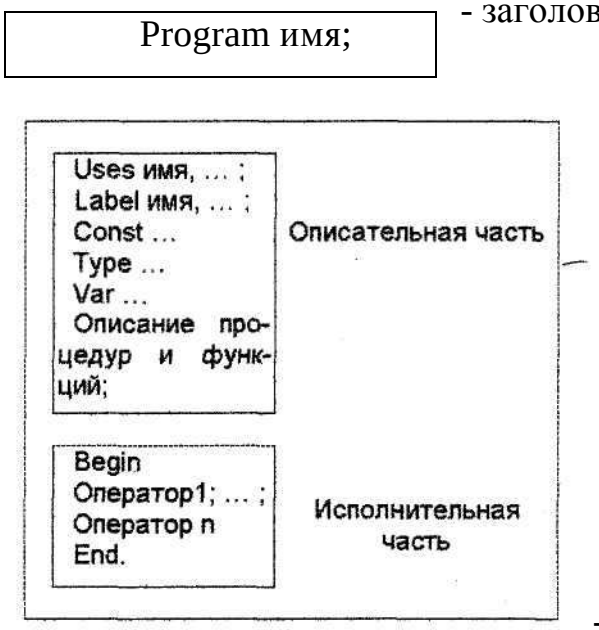
цию программ. В идеальном случае программа на Паскале состоит из подпрограмм (процедур и функций), которые вызываются для выполнения из раздела операторов основной программы.

Программный блок состоит из двух частей: описательной и исполнительной. Описательная часть в общем случае включает в себя 6 разделов: список имён подключаемых модулей (он определяется ключевым словом **Uses**), описание меток, описание констант, определение типов данных, описание переменных, описание процедур и функций. Исполнительная часть (её ещё называют разделом операторов) начинается ключевым словом **Begin** (начало), далее следуют операторы, записанные согласно алгоритму решаемой задачи и отделенные друг от друга точкой с запятой.

Завершается исполнительная часть программного блока ключевым словом **End** (конец) с точкой. Слова **Begin** и **End** являются аналогом открывающей и закрывающей скобок в обычных арифметических выражениях, поэтому их называют ещё **операторными скобками.**

Структуру программы на Паскале в общем случае можно представить

следующем образом:



В программе любой описательный раздел может отсутствовать. Разделы описания могут следовать в любом порядке (кроме Uses, который всегда располагается после заголовка программы). Главное, чтобы все описания элементов были бы сделаны до того, как они будут использоваться.

При компиляции программы процессор ПК рассматривает содержащиеся перед операторами описания переменных и отводит в памяти соответствующие места для размещения каждой из переменных. При выполнении программы во время вычисления значения выражения производятся обращения за значениями переменных в отведённые для них места памяти, а полученное новое значение для переменной помещается в закреплённое за данной переменной место в памяти, а предыдущее значение этой переменной стирается.

Раздел Uses. Он состоит из ключевого слова **Uses** и списка имён подключаемых стандартных и пользовательских модулей.

Пример: Uses Crt, Dos, MyLib;

Раздел описания меток. Перед любым оператором можно поставить метку, состоящую из имени и следующего за ним двоеточия. Именем метки может служить идентификатор или число. Все метки должны быть описаны. Раздел описания меток начинается ключевым словом **Label** (метка), за которым следуют имена меток, разделённые запятыми. За последним именем ставится точка с запятой. Например:

Label Blok, Ml, 5, 15;

Раздел описания констант. В этом разделе производится присвоение идентификаторам констант постоянных значений. Раздел начинается ключевым словом **Const**, за которым следуют выражения, присваивающие идентификаторам (через =) постоянные числовые или строковые значения. Эти выражение отделяются друг от друга точкой с запятой.

Пример. **Const A=50; B2='Блок1';**

Раздел описания типов данных. Тип данных может быть описан либо в разделе описания переменных, либо определяться идентификатором типа. Раздел описания типов данных начинается ключевым словом **Type,** за которым следует определение типов, разделяемых точкой с запятой.

Пример. **Type** Matr = array [1..10] of real;

Dni = 1..31; LatBukva = ('a'..'2');

Раздел описания переменных. Каждая встречающаяся в программе переменная должна быть описана. Раздел описания переменных начинается ключевым словом **Var,** затем через запятые перечисляются имена переменных и через двоеточие указывают их тип, а после типа ставится точка с запятой.

Пример. **Var A,B,C:integer; Res,Sum:real;**

Ll, Vhod:boolean;

Раздел описания процедур и функций. В этом разделе размещаются тела подпрограмм. ***Подпрограммой*** *называется программная единица, имеющая имя, по которому она может быть вызвана из других частей программы*. В Паскале роль подпрограмм выполняют процедуры и функции, которые подразделяются на стандартные и определённые пользователем. Стандартные процедуры и функции являются частью языка Паскаль и могут вызываться без предварительного описания. А процедуры и функции пользователя должны описываться обязательно. В общем случае подпрограмма имеет ту же структуру, что и основная программа. При описании подпрограмм их заголовки начинаются ключевыми словами: **Procedure** или **Function.** Более подробное рассмотрение описания процедур и функций пользователя приведено позже (в лабораторной работе № 8).

Комментарии. ***Комментарий*** *— это пояснительный тест, который можно записать в любом месте программы, где размещён пробел.* Текст комментария ограничивается символами {} или (**) и может использовать любые комбинации латинских и русских букв, цифр и других символов алфавита языка Паскаль. Комментарии игнорируются компилятором.

В процессе отладки программы часто требуется временно исключить выполнение какой-либо части программы. Это удобно выполнить путём заключения временно этой части программы в символы {} или (**), которые после отладки программы можно убрать, и программа будет выполняться в полном объёме.

2.Программирование линейных алгоритмов

Линейный алгоритм использует такие символы-блоки (см. рис.1): *терминатор* (начало, конец), *данные* (ввод/вывод данных), *процесс* (блоки 4-6). Рассмотрим программирование данных символов-блоков на языке Паскаль.

Блок 1 (начало). Этот блок включает заголовок программы на Паскале, все описательные разделы и операторную скобку **Begin** раздела операторов.

Блоки ввода/вывода данных. Эти блоки программируются с помощью операторов ввода/вывода данных: **Read, ReadLn, Write, WriteLn.**

Операторы **Read** и **ReadLn** обеспечивают ввод данных с клавиатуры для последующей их обработки программой. Их форматы:

Read (xl,x2, ...,xn); **ReadLn(**xl,x2,... ,xn); ,

где xl, x2, ..., xn - вводимые переменные. Значения переменных xl,x2, ...,xn при вводе набираются на клавиатуре минимум через один пробел (но не через запятую) и высвечиваются на экране. После набора данных для одного оператора **Read** или **ReadLn** нажимается клавиша **<Enter>.** Значения вводимых переменных должны соответствовать этим переменным по очерёдности и типам. Если соответствие по типам будет нарушено, то возникнет ошибка ввода и появится сообщение об этом. Если соответствие будет нарушено по очерёдности, то будут неверными результаты вычислений по программе.

Если в программе имеется несколько операторов **Read,** данные для них можно набирать в одной строке потоком, так как после считывания значений переменных по первому оператору **Read** курсор остаётся в той же строке вводимых данных.

Оператор ввода **ReadLn** аналогичен оператору **Read,** за исключением того, что после считывания значения последней переменной одного оператора **ReadLn** курсор перейдёт на начало следующей строки и данные для очередного оператора **ReadLn** должны набираться с начала новой строки.

Операторы **Read** и **ReadLn** требуют *обязательного* ввода данных. Если вы их не введёте, а просто нажмёте клавишу **<Enter>,** то работа оператора ввода не закончится и он будет ожидать ввода конкретной информации, а вы не сможете перейти к выполнению следующего оператора программы.

Значения переменных xl,x2,...,xn в операторах **Read** и **ReadLn** при вводе можно набирать и в нескольких строках, нажимая клавишу **<Enter>** после набора каждой строки данных, так как в данном случае после каждого нажатия клавиши **<Enter>** курсор переходит в начало следующей строки.

Оператор **ReadLn;,** записанный в программе без списка водимых переменных, приводит к остановке её дальнейшего выполнения до тех пор, пока вы не нажмёте клавишу **<Enter>.** Поэтому можно использовать оператор **ReadLn;** для просмотра выводимых результатов вычисления по программе, например:

… … …

WriteLn ('x=',x,'y=',y);

Write (‘Нажмите Enter’);

ReadLn;

End.

В Турбо-Паскале допускается вводить значения следующих данных: целых, вещественных, символьных и строковых переменных.

С помощью оператора ввода нельзя ввести:

1) значение логической переменной;
2) значение переменной перечисляемого типа;
3) значения структурированных переменных (массивов, множеств, записей).

Пример 1. **Var** A, B**,** C: real;

I, K: integer;

… … …

ReadLn (A, B, C);

Read (I, K);

В данном примере значения переменных вводятся в следующем порядке:

0.5 6.25 -7.1E-1

1 5

или 0.5

6.25

-7.1E-1

1

5

Но нельзя все числа записать в одной строке, так как используется оператор **ReadLn.** После выполнения операции ввода переменным будут присвоены такие значения: A = 0,5; B = 6,25; C = -0,71; I=1; K = 5.

Пример 2. Пусть имеются переменные следующих типов: R: real; Cl, C2, C3 : char, которым необходимо присвоить соответственно значения: 1,5; 'A' ; 'B' ; 'C. Для этого используется оператор **Read (R,** Cl, C2, C3); .

При вводе значения переменных можно расположить следующим образом:

1.5ABC или **1.5EOABC** (без апострофов), но нельзя после 1.5 поместить пробел, так как он воспримется как значение символьной константы.

Оператор вывода данных имеет формы записи:

1) **Write** (список переменных) – выводит последовательно значения переменных из списка;

2) **WriteLn**(список переменных) – то же, что и оператор Write, но после вывода значения последней переменной из списка осуществляется переход на новую строку;

3) **WriteLn ;** -- осуществляет переход на новую строку (без вывода данных).

Транслятор по умолчанию отводит определенное число позиций для выводимых величин каждого типа. Все элементы вывода печатаются в строку в заданном порядке, при этом пробелы между ними автоматически не ставятся. Их при желании необходимо учитывать самим при программировании операторов вывода.

Пример. Пусть в результате выполнения программы переменные получили такие значения: I=-5, R=3.52, C= '+' , B= True.

Выведем их на печать:

```
Program Pr;
Var I: integer;
R:real;
C:char;
B:Boolean;
Write (‘Пример‘); WriteLn;
WriteLn (’ I=’, I,’ R=’, R’);
WriteLn (‘ C=’, C );
WriteLn (‘ B=’, B );
```

End.

В результате выводимые значения примут вид:

I=-5 R= 5.2000000000 E+00

C=+

B= True.

Форматный вывод данных. В ТП предусмотрен вывод данных с форматами. В общем случае формат имеет вид:

P: M,

где **P**- имя переменной,

M – целая константа, указывающая на число позиций для выводимой величины **P**.

Для вещественных переменных формат может быть задан и в таком виде:

P: M: N,

где **M** –общее число позиций для выводимой переменной **P**, включая знак числа, целую часть, точку и дробную часть;

N – число позиций дробной части. Если парметры M и N опущены, то вещественная переменная выводиться в виде константы с плавающей точкой.

Пример. Используем форматный вывод переменных из предыдущего примера:

WriteLn (‘ I=’, I:3, ’ R=’,R:5:2);

Write (‘ C=’, C:2, ’ B=’,B:6);

В результате получим:

I= 5_R= 3.52

C= + B= True.

Необходимо помнить, что все символы (включая пробелы, запятые), заключенные между открывающим и закрывающим апострофом в списке переменных операторов **Write** и **WriteLn,** выводятся на экран как элементы текста. *Если при выводе значения некоторой переменной выводимое число не будет помещаться в указанный формат, то часть значения переменной, расположенная перед десятичной точкой, будет выведена на экран полностью*. При этом число позиций, предназначенных для вывода дробной части числа остается равным указанной в формате величин (дробная часть, не укладывающаяся в заданное число позиций, округляется; округление не изменяет самого значения переменной, а касается только процесса вывода этого значения).

Если в выводимом числе дробная часть отсутствует, оно выводится в экспоненциальной форме с достаточным для точного изображения числом позиций. Но если же вы хотите выдать значение вещественного числа без дробной части (и без экспоненты), то необходимо указать следующий формат:

WriteLn (P :M :O);

Использование форматного вывода позволяет корректно оформлять различного рода таблицы.

Ввод данных с клавиатуры можно осуществлять и по запросам. В этом случае необходимо запрограммировать соответствующие запросы на ввод данных, используя операторы **Write**, а ввод численных значений - по операторам **Read** или **ReadLn**. Например, запрограммируем ввод переменных x = 37,5; y = $-0,7*10^2$; z = -2,73 по следующим запросам:

Введите значение x = 37.5

1-й запрос ответ пользователя на запрос (набор на клавиатуре);

Введите значение y = -0.7E+02

2-й запрос ответ пользователя на запрос;

Введите значение z = -2.73

3-й запрос ответ пользователя на запрос.

Тогда на Паскале такой ввод данных будет иметь вид:

Write ('Введите значение x = ');

ReadLn (x);

Write ('Введите значение y = ');

ReadLn (y);

Write ('Введите значение z = ');

ReadLn (z);

Блоки 4-6 (рис.1) линейного алгоритма программируются с помощью *операторов присваивания*. Общий вид оператора присваивания следующий:

Идентификатор переменной := выражение;

Оператор присваивания предписывает выполнить *выражение*, записанное в его правой части, и результат вычисления по этому выражению присвоить (**:=**) *переменной*, идентификатор которой расположен в левой части оператора. Допустимо присваивание любых типов данных, кроме *файловых*.

Пример.

Y :=Sqrt (x)+1;

b :=M and N;

В операторе присваивания идентификатор переменной и выражение должны иметь один и тот же тип, кроме одного исключения: *идентификатору переменной типа* **real** *разрешается присваивать выражение типа* **integer**.

Блок 8 (рис.1) линейного алгоритма программируется оператором **End .** (с точкой).

3. Интегрированная среда программирования Турбо-Паскаль

Интегрированная среда программирования Турбо-Паскаль (в дальнейшем ТР) включает в себя: экранный редактор, компилятор, редактор связей и отладчик. ТП позволяет набирать тексты программ с использованием внутреннего редактора текстов, компилировать их, выполнять программы и проводить их отладку. Управление всеми этими функциями возможно и в режиме *меню*, и с помощью соответствующих *функциональных клавиш*. Так, для выбора необходимой функции нужно подвести курсор к требуемой команде и нажать клавишу *ввода* (или *нажать выделенную в команде заглавную букву*).

При возникновении ошибки трансляции ТП автоматически переходит в режим экранного редактирования и ставит курсор в точку возникновения ошибки. Аналогичные действия выполняются и отладчиком при возникновении ошибки во время выполнения программы.

Запуск системы программирования Турбо-Паскаль осуществляется командой **Turbo** , после выполнения которой на экране появляется *главное меню* системы.

Для выхода из ТР можно нажать клавиши «**Alt + X**».

ТП использует следующие основные расширения файлов:

com и e*xe* – выполнимые файлы (программы, готовые для выполнения);

pas – файл с исходным текстом программы на Паскале;

bak – резервная копия *pas*- файла;

tp1 - файл, содержащий стандартные модули ТР.

Для работы с ТР обязательными являются два файла: *Turb.exe* (компилятор с интегрированной средой программирования) и *Turbo.tp1* (библиотека стандартных модулей).

Главное меню (*первая строка экрана*) содержит команды: ***File*** (файл), ***Edit*** (редактор), ***Run*** (выполнение), ***Compile*** (компилирование), ***Options*** (опции), ***Debug*** (отладка), ***Break/Watch*** (прерывание/просмотр). Все они, кроме *Edit* , имеют собственные подменю, а некоторые – и несколько вложенных подменю.

Для входа в главное меню можно нажать клавишу **F10**, для выхода из него - **Esc** .

Команда ***File*** содержит функции, управляющие работой с файлами: *Load* - загрузка файла с диска и переход в режим экранного редактирования; *New* - удаление текущей программы из памяти и очистка экрана; *Save* - сохранение на диске текущего редактируемого файла и продолжение редактирования и др.

Команда ***Edit*** активизирует экранный редактор (эта команда не имеет собственного меню).

Команда ***Run*** объединяет функции и команды, управляющие трассировкой и выполнением программы. В этот режим входят следующие функции:

Run - запуск программы на выполнение (при необходимости выполняется трансляция программы). По завершении работы программы происходит возврат в ТР. Синоним этой функции – *Ctrl+F9* ;

Go to cursor - выполнение программы (без трассировки) от текущей строки (в режиме отладки текущая строка выделяется голубым цветом) до строки, в которой находится курсор. Синоним – *F4*;

Trace into - покомандное выполнение (трассировка) программы. Синоним *F7*;

Step over - пооператорное выполнение программы. Синоним *F8*;

User screen - показ результатов выполнения программы, выведенных на экран. Для возврата достаточно нажать любую клавишу. Синоним - *Alt +F5*;

Команда ***Compile*** содержит команды для управления процессом трансляции программы, например *Compile* - трансляция программы (синоним *Alt +F9*) и др.

Команда ***Options*** обеспечивает управление режимами ТР.

Команда ***Debug*** позволяет определять и изменять значения переменных и используется при отладке программы.

Команда ***Break/Watch*** позволяет управлять точками прерывания и переменными в окне просмотра (в этом окне отражаются текущие значения заданных переменных и выражений; для переключения в окно просмотра из окна редактирования служит клавиша F6).

Литература

1. Вальвачев, А.Н., Крисевич, В.С. Программирование на языке Паскаль для персональных ЭВМ ЕС. – Мн.: Выш. шк., 1989. – 223 с.
2. Паскаль для персональных компьютеров / Ю.С. Бородич [и др.]. – Мн.: Выш. шк., 1991. – 365 с.
3. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – М.: Госстандарт, 1990. – 28 с.
4. Офицеров, Д.В., Старых, В.А. Программирование в интегрированной среде Турбо-Паскаль. – Мн.: Беларусь, 1992. – 240 с.

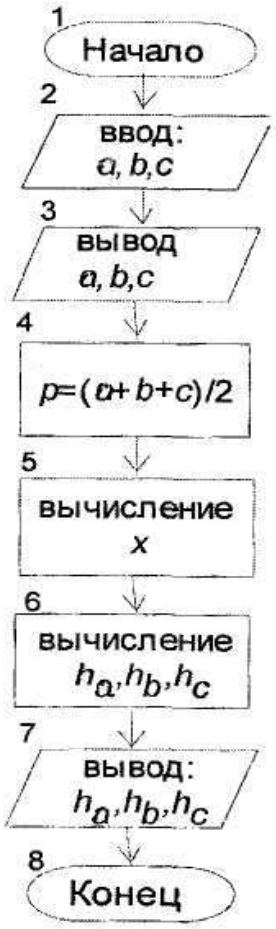


Рис. 1

МОДУЛЬ М4 – «ПРОГРАММИРОВАНИЕ РАЗВЕТВЛЯЮЩИХСЯ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ»

Лабораторная работа № 6

Программирование разветвляющихся вычислительных процессов с использованием условного оператора *IF*

Цель работы*:** *Приобретение практических навыков составления программ решения задач разветвляющейся вычислительной структуры с использованием условного оператора* ***IF .

Постановка задачи

Разработать блок-схему алгоритма и составить Паскаль-программу решения задачи по варианту условия, определяемому номером бригады (табл. 6.1) .

Таблица 6.1

Варианты заданий

№ вариантов	Математические выражения	Исходные данные
1	Y = sin(5k + 3m· | k|); n=1, если k<m; Y = cos(5k + 3m·| k|); n=2, если k>m; Y = k + 5m ; n=3, если k=m	k, m
2	H = arctg(x + |y|); n=1, если x<y; H = arctg(|x| + y); n=2, если x>y; H = $(x + y)^2$; n=3, если x=y	x, y
3	A =$(x + y)^2$ + $\sqrt{x \cdot y}$; n=1, если x·y>0; A =$(x + y)^2$ + $\sqrt{\lvert x \cdot y \rvert}$; n=2, если x·y>0; A =$(x + y)^2$ + 1 ; n=3, если x·y>0;	x, y
4	K = ln(|f| + |g|); n=1, если (f·g)>0; K = e^{f+g} ; n=2, если (f·g)<0; K = f + g; n=3, ес-	f, g

	ли (f·g)=0;	
5	L = $3k^3 + 3p^2$; n=1, если k>|p|; L = |k – p| ; n=2, если k<|p|; L = $(k – p)^2$; n=3, если k=|p|;	k, p
6	C = $x^2 + y^2$ + sin(x); n=1, если x – y = 0; C = $(x - y)^2$ + cos(x); n=2, если x – y > 0; C = $(y – x)^2$ + tg(x); n=3, если x – y < 0;	x, y
7	Y = a + b; n=1, если c = 0; Y = a + b + c; n=2, если c > 0; Y = (a + b) · c; n=3, если c< 0;	a, b, c
8	Y = tg(2x) + z; n=1, если z > 0; Y = $5x^4 + 3x^3 – 2x^2 + 1,5$ + ln|z| ; n=2, если z < 0; Y = sin(x); n=3, если z = 0;	x, z

Содержание лабораторной работы

Лабораторная работа включает:

1. Ознакомление с теоретическими сведениями (см. Приложение 4).

2. Оформление отчета по лабораторной работе, который должен содержать следующие пункты:

1) номер и название работы;

2) цель работы;

3) постановку задачи;

4) блок-схему алгоритма;

5) таблицу идентификаторов;

6) текст исходной Паскаль-программы.

Порядок выполнения работы

Последовательность выполнения работы следующая:

1. Набрать на клавиатуре текст Паскаль-программы .

2. Произвести компиляцию исходной программы.

3. В случае обнаружения ошибок отредактировать программу с ее последующей повторной компиляцией.

2. Запустить программу после сообщения об ее успешной компиляции.

3. Ввести исходные данные для получения окончательного результата, при этом исходные данные подобрать так, чтобы результаты получались по всем ветвям алгоритма.

5. Распечатать текст Паскаль-программы и результаты.

Контрольные вопросы

1. Что Вы понимаете под термином «разветвляющаяся вычислительная структура»?

2. Как строится схема алгоритма разветвляющейся вычислительной структуры?

3. Какой символ осуществляет проверку некоторых условий?

4. Какой оператор Паскаля соответствует этому символу?

5. От чего зависит количество ветвей в алгоритме?

6. Какие структуры оператора If на Паскале Вам известны? Как они выполняются?

Содержание отчета

Отчет по выполненной работе (кроме протокола) должен содержать следующие сведения:

1. Экспериментальные результаты в виде распечатки текста отлаженной Паскаль-программы и результатов счета (по всем ветвям алгоритма).

2. Выводы по работе.

Лабораторная работа № 7

Программирование разветвляющихся вычислительных процессов с использованием оператора выбора *CASE*

Цель работы*:** *Приобретение практических навыков составления программ решения задач разветвляющейся вычислительной структуры с использованием оператора выбора* ***CASE.

Постановка задачи

Построить блок-схему алгоритма и составить Паскаль-программу решения задачи с использованием оператора выбора Case по варианту условия, определяемому номером бригады (табл. 6.1). При этом сначала определяйте значение селектора **n** в зависимости от логического условия, а затем определяйте значение первого математического выражения по оператору выбора ***CASE*** (с селектором **n**).

Содержание лабораторной работы

Лабораторная работа включает:

1. Ознакомление с теоретическими сведениями (см. Приложение 4).

2. Оформление отчета по лабораторной работе, который должен содержать следующие пункты:

1) номер и название работы;
2) цель работы;
3) постановку задачи;
4) блок-схему алгоритма;
5) таблицу идентификаторов;
6) текст исходной Паскаль-программы.

Порядок выполнения работы

Последовательность выполнения работы следующая:

1. Набрать на клавиатуре текст Паскаль-программы .
2. Произвести компиляцию исходной программы.
3. В случае обнаружения ошибок отредактировать программу с ее последующей повторной компиляцией.
4. Запустить программу после сообщения об ее успешной компиляции.
5. Ввести исходные данные для получения окончательного результата, при этом исходные данные подобрать так, чтобы результаты получались по всем ветвям алгоритма.
6. Распечатать текст Паскаль-программы и результаты.

Контрольные вопросы

1.Какова структура оператора выбора ***CASE*** ?

2.Что такое селектор? Как он задается?

3.Какова последовательность работы оператора ***CASE*** ?

Содержание отчета

Отчет по выполненной работе (кроме протокола) должен содержать следующие сведения:

1. Экспериментальные результаты в виде распечатки текста отлаженной Паскаль-программы и результатов счета (по всем ветвям алгоритма).

2. Выводы по работе.

Приложение 4(к модулю М4)

Теоретические сведения к лабораторным работам:

№ 6 *(Программирование разветвляющихся алгоритмов с использованием условного оператора* If *)* ***и № 7*** *(Программирование разветвляющихся алгоритмов с использованием оператора выбора* Case *)*

Программирование разветвляющихся алгоритмов

В разветвляющихся алгоритмах кроме блоков, применяемых в линейных алгоритмах (программирование которых мы рассмотрели в Приложении 3 для рис.1), применяются управляющие блоки «решение», определяющие нужную ветвь дальнейших вычислений в зависимости от выполнения или невыполнения конкретных условий. Так как при этом нарушается естественный порядок выполнения блоков алгоритма, то при программировании разветвляющихся алгоритмов могут использоваться условные (**If** или **Case**) и безусловные (**Goto**) операторы управления. Рассмотрим сначала эти операторы.

Оператор безусловного перехода **Goto** означает « перейти к » и применяется в случаях, когда после выполнения некоторого оператора необходимо выполнять дальше не следующий по порядку оператор, а какой-либо другой, помеченный меткой.

Формат написания оператора безусловного перехода следующий:

Goto Метка;

В соответствии с принципами структурного программирования этот оператор следует применять как можно реже, так как его частое употребление усложняет понимание логики программы.

Условный оператор If (*если*). Он может использоваться в двух формах:

1. **If** B **then** P1 **else** P2 ;

Здесь ключевые слова **If, then , else** означают соответственно: *если, то, иначе;* **B** - любое логическое (булевское) выражение; **P1, P2** - операторы, которые могут быть простыми, составными или условными.

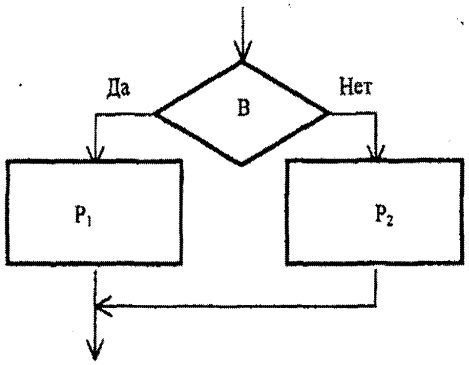


Рис. 2.

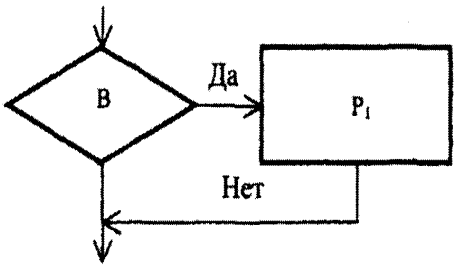


Рис. 3

Управление по этому оператору осуществляется следующим образом (рис. 2): если выражение **B** *истинно*, то выполняется оператор **P1**, а оператор **P2** пропускается; если **B** *ложно*, то пропускается оператор **P1**, а оператор **P2** выполняется.

2. **If** B **then** P1;

Управление по этому оператору осуществляется следующим образом (рис. 3): если выражение B *истинно*, то выполняется оператор **P1** и далее оператор, записанный после оператора **If**; если **B** *ложно*, то оператор **P1** пропускается (не выполняется).

Один оператор **If** может входить в состав другого оператора **If**. В таком случае говорят о вложенности операторов **If**:

If Bl **then If** B2 **then** P1 **else** P2 ;

При вложенности операторов If каждое **else** соответствует тому **then** , которое непосредственно ему предшествует (рис. 4). Конструкций со степе-

нью вложения более 2-3 стараются избегать (из-за сложности их анализа при отладке программы).

Простыми операторами *называются такие операторы, которые не содержат в себе никаких других операторов.* К ним относятся операторы присваивания, безусловного перехода, вызова процедуры (этот оператор рассмотрен позже - в Приложении 7) и пустой оператор.

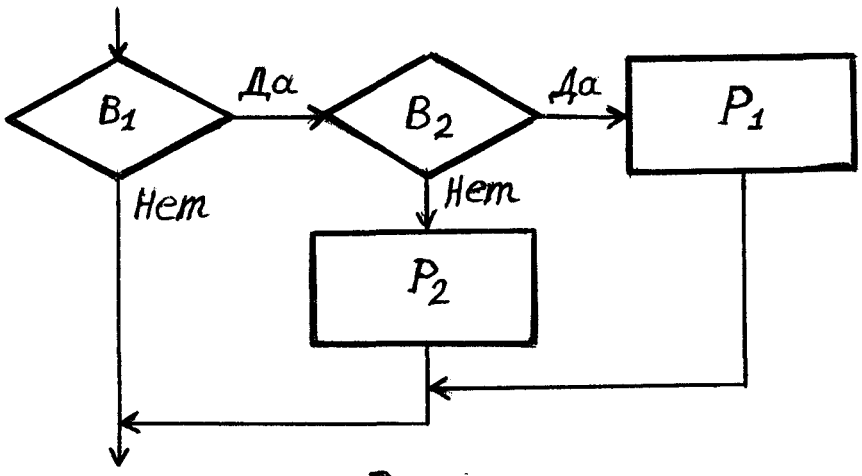


Рис. 4

Пустой оператор *не содержит в себе никаких символов и не выполняет никаких действий.* Он может быть расположен в любом месте программы, где синтаксис языка допускает наличие оператора. Как и все другие операторы, пустой оператор может быть помечен меткой. Чаще всего пустой оператор используется для организации выхода из середины программы или составного оператора:

```
Begin  … ..
Goto Ml; {Переход в конец программы}
… .
M1:      {Пустой оператор с меткой Ml}
End.
```

Составной оператор - это группа из произвольного числа операторов, отделенных друг от друга точкой с запятой и ограниченных операторными

скобками **begin** и **end** . Составной оператор воспринимается как единое целое и может находиться в любом месте программы (чаще всего он используется в условных операторах и операторах повтора).

Оператор выбора Case. Этот оператор является обобщением оператора **If** и позволяет сделать выбор из произвольного числа вариантов. Он состоит из *выражения* - *селектора* и *последовательности операторов*, каждому из которых предшествует *список констант выбора* (список может состоять и из одной константы). Как и оператор If, оператор **Case** может использоваться в двух формах: со словом **else** , имеющим тот же смысл, как и в операторе **If**, и без него. Их форматы записи следующие:

1. **Case** К **of**
   S1: P1;
   S2: P2;
   …
   Sn: Pn
   else Pn+1
   end;
2. **Case** К of
   SI:PI :
   S2: P2;
   …
   Sn: Pn
   end;

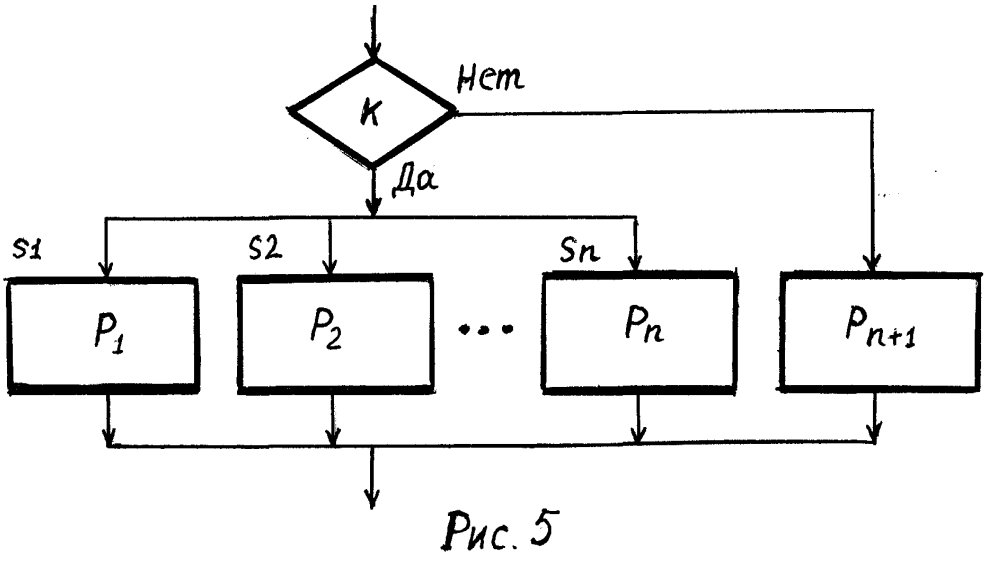


Рис. 5

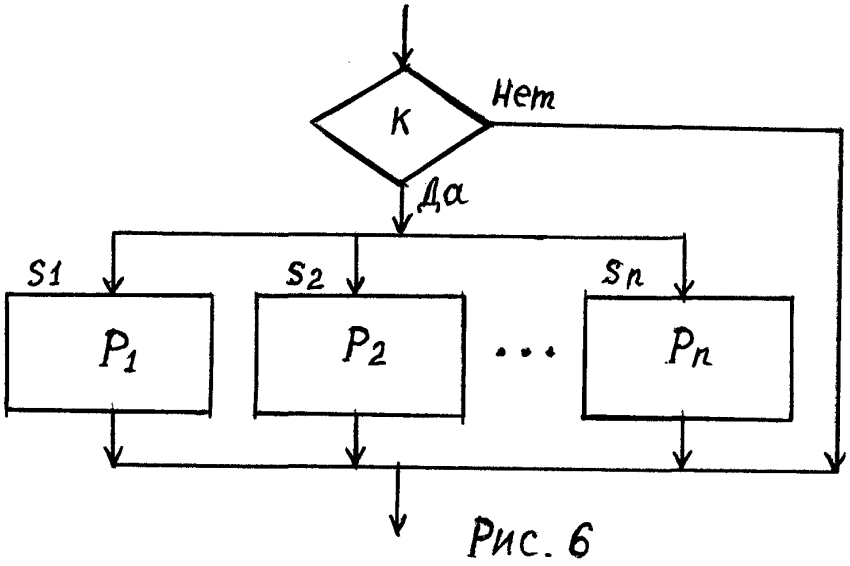


Рис. 6

Здесь **К** - *выражение-селектор*, которое может быть одним из скалярных типов (кроме **real**), т.е. целого, логического, символьного или пользовательских типов.

Оператор **Case** работает следующим образом (рис. 5): сначала выполняется тот из операторов **Pi**, константа выбора которого равна текущему значению *выражения-селектора* **К**. Если ни одна из констант **Si** не равна текущему значению **К**, то выполняется оператор, стоящий за словом **else.** Во втором случае (рис. 6), когда слово **else** в операторе **Case** отсутствует, если ни одна из констант **Si** не равна текущему значению **К**, то выполняется первый оператор, за границей **end;** (оператор **Case** в этом случае пропускается).

Список констант выбора состоит из произвольного количества значений или диапазонов. Тип констант должен совпадать с типом выражения-селектора **К**.

Пример. Составим программу разветвляющегося алгоритма (рис. 7) по вычислению функции

$$D = \begin{cases} Ln(ab)^2, & ab<0 \\ Ln(ab), & ab>0 \\ (a+b), & ab=0. \end{cases}$$

Пусть исходные переменные a и b вещественного типа. Запрограммируем, например, разветвляющуюся структуру с блоком «решение» 3 с помощью оператора **If**, а - с блоком «решение» 5 с помощью оператора **Case** . Тогда программа будет иметь вид:

```
Program RazAlg;
 Var a, b, d :real;
  Begin
   ReadLn (a, b);
   If a*b<0 then
    D := Ln (sqr (a*b))
    else Case a*b>0 of
        True: D:=Ln(a*b)
        else D:= a+b
        end;
   WriteLn(' D = ',D);
   End.
```

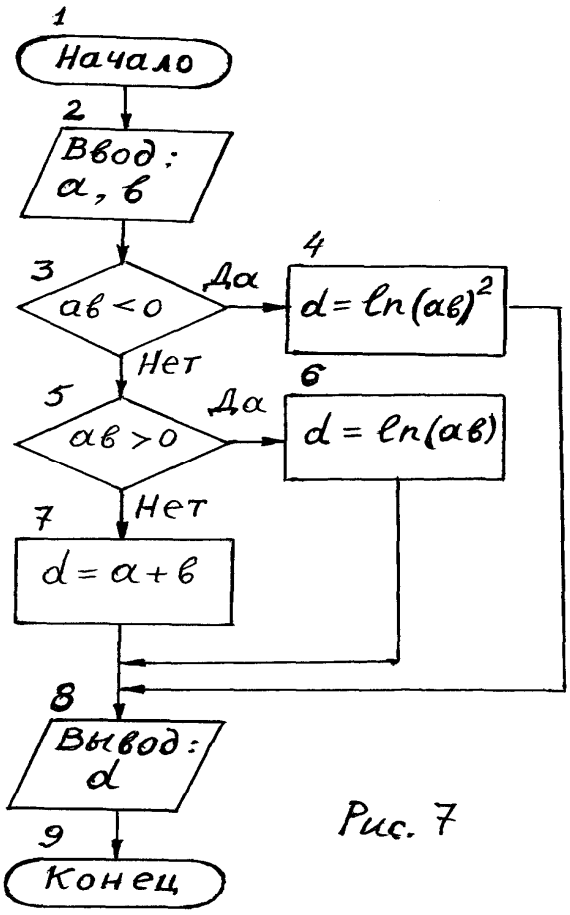


Рис. 7

Литература

5. Вальвачев, А.Н., Крисевич, В.С. Программирование на языке Паскаль для персональных ЭВМ ЕС. – Мн.: Выш. шк., 1989. – 223 с.

6. Паскаль для персональных компьютеров / Ю.С. Бородич [и др.]. – Мн.: Выш. шк., 1991. – 365 с.

7. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – М.: Госстандарт, 1990. – 28 с.

8. Офицеров, Д.В., Старых, В.А. Программирование в интегрированной среде Турбо-Паскаль. – Мн.: Беларусь, 1992. – 240 с.

МОДУЛЬ М5 – «ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКИХ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ»

Лабораторная работа № 8

ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКИХ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ С ИСПОЛЬЗОВАНИЕМ ОПЕРАТОРА ЦИКЛА *FOR*

Цель работы*:** Приобретение практических навыков составления программ решения задач, содержащих циклические вычислительные структуры, с использованием оператора цикла ***FOR .

Постановка задачи

Разработать блок-схему алгоритма и составить Паскаль-программу решения задачи с использованием оператора цикла ***FOR*** по варианту условия, определяемому номером бригады (табл. 8.1) .

Таблица 8.1

Варианты заданий

№ вариантов	Математические выражения	Исходные данные
1	$Y = \sin(x) + \sin^2(x) + \ldots + \sin^{15}(x)$	x
2	$H = 1/a + 1/a^2 + \ldots + 1/a^{25}$	a
3	$A = 1 - x + x^2/2! - x^3/3! + \ldots + x^{12}/12!$	x
4	$K = \cos(x) + \cos(x^2) + \ldots + \cos(x^{30})$	x

5	$L = x + x^3/3! + x^5/5! + \dots x^{15}/15!$	x
6	$C = \sum_{i=1}^{10} (x/i! + \sqrt{\lvert x \rvert})$	x
7	$S = \prod_{i=1}^{15} (x/i)$	x
8	$M = \sum_{i=1}^{10} (x/i^2)$	x

Содержание лабораторной работы

Лабораторная работа включает:

1. Ознакомление с теоретическими сведениями (см. Приложение 4).

2. Оформление отчета по лабораторной работе, который должен содержать следующие пункты:

1) номер и название работы;
2) цель работы;
3) постановку задачи;
4) блок-схему алгоритма;
5) таблицу идентификаторов;
6) текст исходной Паскаль-программы.

Порядок выполнения работы

Последовательность выполнения работы следующая:

1.Набрать на клавиатуре текст Паскаль-программы .

2. Произвести компиляцию исходной программы.

3. В случае обнаружения ошибок отредактировать программу с ее последующей повторной компиляцией.

4.Запустить программу после сообщения об ее успешной компиляции.

5.Ввести исходные данные для получения окончательного результата.

6.Распечатать текст Паскаль-программы и результаты.

Контрольные вопросы

1.Какова структура оператора ***FOR*** *?* Как он работает *?*

2.Как записывается оператор ***FOR*** , если он охватывает группу операторов ?

3.Какими операторами можно запрограммировать циклический вычислительный процесс с из вестным числом повторений цикла ?

4.Как программируются циклические вычислительные процессы с неизвестным числом повторений цикла ?

Содержание отчета

Отчет по выполненной работе (кроме протокола) должен содержать следующие сведения:

1. Экспериментальные результаты в виде распечатки (или запись с экрана дисплея) текста отлаженной Паскаль-программы и результатов счета .

2. Выводы по работе.

Лабораторная работа № 9

ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКИХ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ С ИСПОЛЬЗОВАНИЕМ ОПЕРАТОРА ЦИКЛА С ПРЕДУСЛОВИЕМ *While*

Цель работы*:** Приобретение практических навыков составления программ решения задач, содержащих циклические вычислительные структуры, с использованием оператора цикла ***While .

Постановка задачи

Разработать блок-схему алгоритма и составить Паскаль-программу решения задачи с использованием оператора цикла ***While*** по варианту условия, определяемому номером бригады (табл. 9.1) .

Таблица 9.1

Варианты заданий

№ вариантов	Математические выражения	Изменяемые параметры	Исходные данные
1	$S = \frac{ax^2 + \sin^2 z}{\sqrt{1+e^y}}$	Параметр ***x*** изменяется от ***x*** = $x_н$= 1 до ***x*** = $x_к$ = 4,5 с шагом h = 0,5	a, z, y – константы, значения которых задать самостоятельно
2	$M = \frac{\beta^2 + \sqrt{\lvert q \rvert}}{\cos^2 x + \beta \ln y}$	Параметр ***x*** изменяется от ***x*** = $x_н$= 1 до ***x*** = $x_к$ = 5 с шагом h = 0,5	β, q, y – константы, значения которых задать самостоятельно
3	$W = \frac{\sin^2 (z+a)^3}{t\sqrt[3]{e^{a+2q}}}$	Параметр z изменяется от z = $z_н$= 0,3 до z = $z_к$ = 1 с шагом h = 0,1	*a*, z, y – константы, значения которых задать самостоятельно
		Параметр ***x*** из-	α, q, t, y –

4	$K = \frac{3x^2 - \sqrt{\cos(q^3)}}{\ln^2(y+\alpha)t}$	меняется от $\boldsymbol{x} = \boldsymbol{x}_н = 0{,}2$ до $\boldsymbol{x} = \boldsymbol{x}_к = 1{,}5$ с шагом $h = 0{,}1$	константы, значения которых задать самостоятельно
5	$L = \frac{4\delta\sqrt{\lvert x + \sin(z^3) \rVert}}{3\ln^2(q+x)}$	Параметр z изменяется от $z = z_н = 0{,}3$ до $z = z_к = 1{,}5$ с шагом $h = 0{,}1$	δ, x, q – константы, значения которых задать самостоятельно
6	$C = \frac{a\sqrt[3]{x + \ln^2 y}}{\lvert t^3 \rvert}$	Параметр y изменяется от $y = y_н = 0{,}2$ до $y = y_к = 1{,}5$ с шагом $h = 0{,}2$	a, x, t – константы, значения которых задать самостоятельно
7	$N = \frac{p^3 + e^{2\beta t}}{13{,}2\sqrt{\ln(\alpha + t)}}$	Параметр t изменяется от $t = t_н = 1$ до $t = t_к = 7$ с шагом $h = 1$	p, β, α – константы, значения которых задать самостоятельно
8	$P = \frac{\lvert \alpha^3 + \sqrt[3]{\sin^2 z} \rvert}{\sqrt{x}e^{\alpha t}}$	Параметр z изменяется от $z = z_н = 0{,}5$ до $z = z_к = 4{,}5$ с шагом $h = 0{,}5$	α, x, t – константы, значения которых задать самостоятельно

Содержание лабораторной работы

Лабораторная работа включает:

1. Ознакомление с теоретическими сведениями (см. Приложение 5).

2. Оформление отчета по лабораторной работе, который должен содержать следующие пункты:

1) номер и название работы;
2) цель работы;
3) постановку задачи;
4) блок-схему алгоритма;
5) таблицу идентификаторов;
6) текст исходной Паскаль-программы.

Порядок выполнения работы

Последовательность выполнения работы следующая:

1. Набрать на клавиатуре текст Паскаль-программы .

2. Произвести компиляцию исходной программы.

3. В случае обнаружения ошибок отредактировать программу с ее последующей повторной компиляцией.

4. Запустить программу после сообщения об ее успешной компиляции.

5.Ввести исходные данные для получения окончательного результата .

6.Распечатать текст Паскаль-программы и результаты.

Контрольные вопросы

1. Какие типы циклов встречаются в циклических вычислительных процессах ?
2. Какова структура оператора цикла ***While*** ? Как он работает ?
3. Как осуществляется в операторе ***While*** выход из цикла?

Содержание отчета

Отчет по выполненной работе (кроме протокола) должен содержать следующие сведения:

1. Экспериментальные результаты в виде распечатки текста отлаженной

Паскаль-программы и результатов счета .

2. Выводы по работе.

Лабораторная работа № 10

ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКИХ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ С ИСПОЛЬЗОВАНИЕМ ОПЕРАТОРА ЦИКЛА С ПОСТУСЛОВИЕМ *Repeat*

Цель работы: Приобретение практических навыков составления программ решения задач, содержащих циклические вычислительные структуры, с использованием оператора цикла ***Repeat***

Постановка задачи

Разработать блок-схему алгоритма и составить Паскаль-программу решения задачи с использованием оператора цикла ***Repeat*** по варианту условия, определяемому номером бригады (табл. 9.1) .

Содержание лабораторной работы

Лабораторная работа включает:

1. Ознакомление с теоретическими сведениями (см. Приложение 5).

2. Оформление отчета по лабораторной работе, который должен содержать следующие пункты:

1) номер и название работы;
2) цель работы;
3) постановку задачи;
4) блок-схему алгоритма;
5) таблицу идентификаторов;
6) текст исходной Паскаль-программы.

Порядок выполнения работы

Последовательность выполнения работы следующая:

1. Набрать на клавиатуре текст Паскаль-программы .
2. Произвести компиляцию исходной программы.
3. В случае обнаружения ошибок отредактировать программу с ее последующей повторной компиляцией.
4. Запустить программу после сообщения об ее успешной компиляции.
5. Ввести исходные данные для получения окончательного результата .
6. Распечатать текст Паскаль-программы и результаты.

Контрольные вопросы

1. Какова структура оператора цикла ***Repeat*** ? Как он работает ?
2. Чем отличаются между собой операторы цикла ***Repeat*** и ***While***?
3. Какие служебные слова в операторе ***Repeat*** обозначают границы тела цикла ?

Содержание отчета

Отчет по выполненной работе (кроме протокола) должен содержать следующие сведения:

1. Экспериментальные результаты в виде распечатки текста отлаженной Паскаль-программы и результатов счета .
2. Выводы по работе.

Приложение 5 ***(к модулю М5)***

Теоретические сведения к лабораторным работам:

№ 8 **- «ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКИХ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ С ИСПОЛЬЗОВАНИЕМ ОПЕРАТОРА ЦИКЛА** ***FOR*** **», *№ 9*** **- «ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКИХ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ С ИСПОЛЬЗОВАНИЕМ ОПЕРАТОРА ЦИКЛА С ПРЕДУСЛОВИЕМ *While*», *№ 10*** **– «ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕ-**

СКИХ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ С ИСПОЛЬЗОВАНИЕМ ОПЕРАТОРА ЦИКЛА С ПОСТУСЛОВИЕМ *Repeat*»

Программирование циклических алгоритмов

При программировании циклических алгоритмов используются операторы повтора: **For** (для), **Repeat** (повторять) и **While** (пока).

Оператор повтора **For** состоит из заголовка и тела цикла. Тело цикла представляет собой один или несколько операторов (согласно алгоритму), которые могут выполняться более одного раза. Оператор **For** используется обычно тогда, когда количество повторов известно заранее. Он может представляться в двух формах:

1. **For** I**:=** nl **to** n2 **do** P**;**
2. **For** I:= nl **downto** n2 **do** P**;** ,

где nl, n2 - *выражения*, определяющие начальное и конечное значения параметра цикла **I**;

For ... **do** - заголовок цикла;

P - тело цикла, которое может быть простым или составным оператором.

Оператор **For** обеспечивает выполнение цикла до тех пор, пока не будет перебраны все значения параметра цикла **I** от начального до конечного значения включительно (рис.8). Параметр цикла, его начальное и конечное значения должны принадлежать к одному и тому же типу, при этом допустим любой скалярный тип, кроме вещественного. Если используется тип integer, byte и интервальный, то значения параметра цикла последовательно увеличиваются (при **For** ... **to)** или уменьшаются (при **For** ... **downto)** на единицу при каждом повторе.

В теле цикла нельзя использовать операторы, изменяющие значение параметра цикла. После нормального (если не используется преждевременный выход из цикла с помощью оператора **goto**) завершения оператора значение параметра цикла равно его конечному значению.

В теле оператора For могут находиться другие операторы **For** (при вложенных циклах, рис. 9) :

Оператор повтора **Repeat** записывается в виде :

Repeat P1; …; Pn **until** B ; ,

где операторы P1; ... ; Pn - тело цикла, B - выражение булевского типа.

Этот оператор означает: *повторять* (**Repeat**) выполнять тело цикла *пока не* (**until**) станет выражение B истинным, после чего осуществляется выход из цикла (рис. 10).

Оператор **Repeat** имеет такие характерные особенности : тело цикла выполнятся по крайней мере один раз; оно выполняется до тех пор, пока условие B равно **False**; в теле цикла может находиться несколько операторов, при этом операторные скобки **begin** и **end** не используются; по крайней мере один оператор из операторов тела цикла должен влиять на значение условия B, а иначе цикл может повторяться бесконечно.

Пример. Программа вычисления суммы четных чисел в интервале (0 - 10) включительно согласно алгоритму рис. 11, составленная с использованием оператора **Repeat**, имеет вид:

```
Program CklAlg ;
Var I, S : integer ;
Begin
   I : = 0 ;  S : = 0 ;
    Repeat  S : = S + I ;
            I : = I + 2
            Until ( I > 10 ) ;
     Writeln ( ‘ S = ‘ , S )
End .
```

Оператор **While** похож на оператор **Repeat** , но проверка логического условия **B** выполнения тела цикла осуществляется в самом начале оператора. Он записывается в виде:

While B **do** P ;

и означает: *пока* (While) выражение B истинно, то необходимо *выполнять* (do) тело цикла P (рис. 12).

Перед каждым выполнением тела цикла вычисляется значение выражения **B**. Если результат равен **True** , происходит выполнение тела цикла **P**, а если значение выражения **B** равно **False** , то тело цикла **P** не выполняется и осуществляется переход к первому после цикла оператору. Но ели значение выражения **B** равно **False** , то тело цикла не выполняется ни разу и происходит *сразу* переход к следующему после цикла оператору.

Как и в операторе Repeat, в теле цикла должен присутствовать оператор, изменяющий переменную, входящую в условие B, иначе цикл может оказаться бесконечным.

Оператор While, как и операторы For и Repeat, может быть вложенным.

Пример. Программа, в которой с помощью оператора While вычисляется значение **n!**, имеет вид:

```
Program Fakt ;
Var n , I , p : integer ;
 Begin
    Readln (n) ;
     P : = 1 ; I : = 0 ;
     While I < n do
         Begin
            I : = I + 1 ;
            P : = p * I
          End ;
```

```
Write ( ‘ n! = ‘ , p, ‘    n = ‘, n )
End .
```

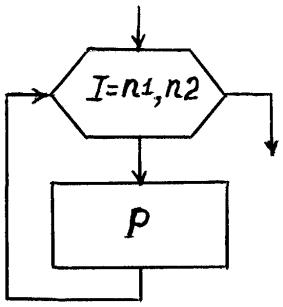


Рис. 8

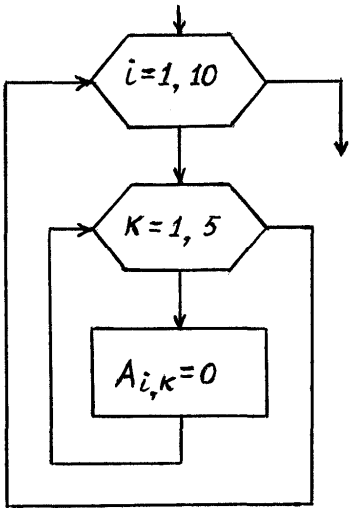


Рис. 9

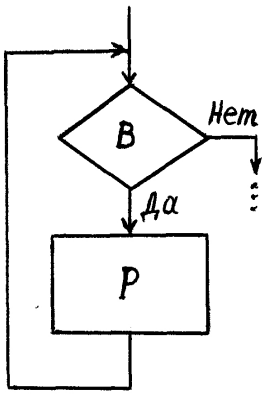


Рис. 12

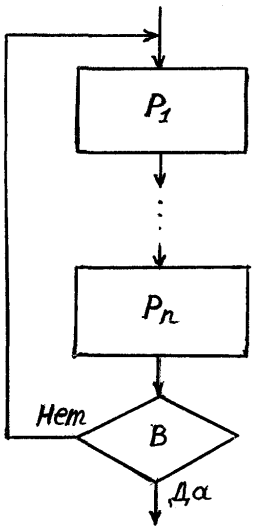


Рис. 10

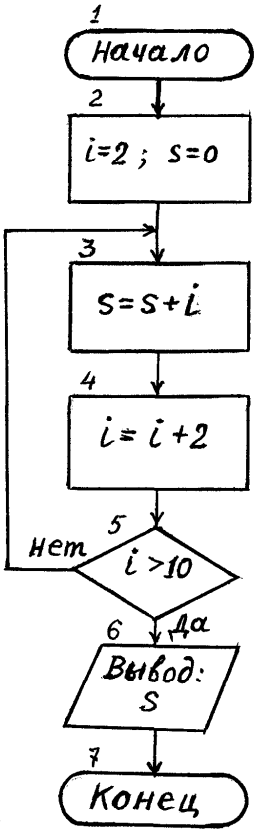


Рис. 11

Литература

1. Вальвачев, А.Н., Крисевич, В.С. Программирование на языке Паскаль для персональных ЭВМ ЕС. – Мн.: Выш. шк., 1989. – 223 с.
2. Паскаль для персональных компьютеров / Ю.С. Бородич [и др.]. – Мн.: Выш. шк., 1991. – 365 с.
3. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – М.: Госстандарт, 1990. – 28 с.
4. Офицеров, Д.В., Старых, В.А. Программирование в интегрированной среде Турбо-Паскаль. – Мн.: Беларусь, 1992. – 240 с.