**Введение.**

Язык Паскаль был создан как учебный язык программирования в 1968 –1971г. Никлаусом Виртом. В настоящее время этот язык имеет более широкую сферу применения, чем предусматривалось при его создании. Целью работы Вирта было создание языка, который:

* Строился бы на небольшом количестве базовых понятий;
* Имел бы простой синтаксис;

Допускал бы перевод программ в машинный код простым компилятором;

Все эти качества сделали язык очень популярным и удобным для применения в школе.

Знакомясь с языком Паскаль я выбрал для своей работы конструкцию ветвления как наиболее распространённую и интересную.

### Цель моей работы – изучить конструкцию ветвления и её применение в языке программирования Паскаль. Исходя из этого я поставил следующие задачи:

1. Изучить литературу по данной теме.
2. Составить план своей работы.
3. Изучить алгоритмическую конструкцию ветвления.
4. Рассмотреть её применение в Паскале.
5. Научиться решать задачи с Ветвлением.
6. Составить пакет заданий по теме “Ветвление в Паскале”.

**I. Ветвление в алгоритмах.**

**Блок – схемы и словесное описание ветвлений.**

**1.1 Основные этапы решения задач на компьютере.**

Процесс решения задач на компьютере – это совместная деятельность человека и ЭВМ. Этот процесс можно представить виде нескольких последовательных этапов. На долю человека приходятся этапы, связанные с творческой деятельностью – постановкой, алгоритмизацией, программированием задач анализом результатов, а на долю компьютера - этапы обработки информации с соответствии с разработанным алгоритмом.

*Первый этап – постановка задачи*. На этом этапе участвует человек, хорошо представляющий предметную область задачи. Он должен чётко определить цель задачи, дать словесное описание содержания задачи и предложить общий подход к её решению. Для вычисления суммы двух целых чисел человек, знающий, как складываются числа, может описать задачу следующим образом: ввести два целых числа, сложить их и вывести сумму в качестве результата решения задачи.

*Второй этап – математическое и информационное моделирование.* Цель этого этапа – создать такую математическую модель решаемой задачи, которая может быть реализована в компьютере. Существует целый ряд задач, где математическая постановка сводится к простому перечислению формул и логических условий. Этот этап тесно связан с первым этапом, и его можно отдельно не рассматривать, однако возможно, что для полученной модели известны несколько методов решения, и тогда предстоит выбрать лучший.

Для вышеописанной задачи данный этап сведётся к следующему: введённые в компьютер числа запомним в памяти под именами А и В, затем вычислим значение этих чисел по формуле А+В, и результат запомним в памяти под именем Summa.

*Третий этап – алгоритмизация задачи*. На основе математического описания необходимо разработать алгоритм решения.

*Четвёртый этап – программирование*. Программой называется план действий, подлежащий выполнению некоторым исполнителем, в качестве которого может выступать компьютер. Составление программы обеспечивает возможность выполнение алгоритма и соответственно поставленной задачи исполнителем – компьютером. Во многих задачах при программирование на алгоритмическом языке часто пользуются заменой блока алгоритма на один или несколько операторов, введением новых блоков, замена одних блоков на другими.

*Пятый этап – ввод программы и исходных данных в ЭВМ*. Программа и исходные данные вводятся в ЭВМ с клавиатуры с помощью редакторов текстов, и для постоянного хранения осуществляется их запись на гибкий или жёсткий магнитный диск.

*Шестой этап – тестирование и отладка программы*. На этом этапе происходит исполнение алгоритма с помощью ЭВМ, поиск и исключение ошибок. При этом программисту приходится выполнять рутинную работу по проверке работы программы, поиску и исключению ошибок, и поэтому для сложных программ этот этап часто требует гораздо больше времени и сил, чем написание первоначального текста программы.

*Отладка программы* – сложный и нестандартный процесс. Исходный план отладки заключается в том, чтобы оттестировать программу на контрольных примерах.

Контрольные примеры стремятся выбрать так, чтобы при работе с ними программа прошла все основные пути блок – схемы алгоритма, поскольку на каждом из путей могут быть свои ошибки, а детализация плана зависит от того, как поведёт себя программа на этих примерах: на одном может зациклиться (т.е. бесконечно повторять одно и то же действие); на другом – дать явно неверный или бессмысленный результат и т.д. Сложные программы отлаживаются отдельными фрагментами.

Для повышения качества выполнения этого этапа используются специальные программы – отладчики, которые позволяют исполнить программу “по шагам” с наблюдением за изменением значений переменных, выражений и других объектов программы, с отслеживанием выполняемых операторов.

*Седьмой этап – исполнение отлаженной программы и анализ результатов*. На этом этапе программист запускает программу и задаёт исходные данные, требуемые по условию задачи.

Полученные в результате решения выходные данные анализируются постановщиком задачи, и на основе этого анализа вырабатываются соответствующие решения, рекомендации, выводы. Например, если при решение задачи на компьютере результат сложения двух чисел 2 и 3 будет 4, то следует сделать вывод о том, что надо изменить алгоритм и программу.

Возможно, что по итогам анализа результатов потребуется пересмотр самого подхода к решению задачи и возврат к первому этапу для повторного выполнения всех этапов с учётом приобретённого опыта. Таким образом, в процессе создания программы некоторые этапы будут повторяться до тех пор, пока мы получи алгоритм и программу, удовлетворяющие показанным выше свойствам.

**1.2 Алгоритм.**

Слово *алгоритм* происходит от *algorithmic –* латинской формы написания имени латинского математика IX в. Аль-Хорезми, который сформулировал правила выполнения четырёх арифметических действий над многозадачными числами. В дальнейшем алгоритм стали называть описание любой последовательности действий, которую следует выполнить для решения заданной задачи.

Алгоритм может быть ориентирован на исполнение его человеком или автоматическим устройством. Алгоритмы, предназначены для выполнения компьютерами, обычно называют *компьютерными программами* или просто *программами*.

Алгоритмом называется точное предписание, определяющее последовательность действий исполнителя, направленных на решение поставленной задачи. В роли исполнителя алгоритмов могут выступать люди, роботы, компьютеры.

Используются разные способы записи алгоритмов. Широко распространенный словесный способ записи: это записи рецептов приготовления различных блюд в кулинарной книге, инструкции по использованию технических устройств, правила правописания и многие другие. Наглядно представляется алгоритм языком блок-схем.

Например алгоритм решения задачи вычисления суммы двух чисел на языке блок-схем будет записан, как показано на рисунке:

А, В

А+В=С

С

*Свойства алгоритма*. При составлении алгоритма необходимо обеспечить, чтобы он обладал рядом свойств*.*

*Однозначность алгоритма*, под которой понимается единственность толкования исполнителем правил выполнения действий и порядка их выполнения. Чтобы алгоритм обладал этим свойством, он должен быть записан командами из системы команд исполнителя.

Для нашего примера исполнитель алгоритма должен понимать такую запись действий, как сложность числа А и В.

*Конечность алгоритма* – обязательность завершения каждого из действий, составляющих алгоритм, и завершимость выполнения каждого алгоритма в целом. Записанный на рисунке алгоритм обладает этим свойством, так как запись действий исполнителя завершается записью об окончании алгоритма.

*Результативность алгоритма*, предполагающая, что выполнение алгоритма должно завершится получением определенных результатов. Алгоритм в нашем примере обладает этим свойством, так как для целых чисел А и В всегда будет вычислена сумма.

*Массовость*, т.е. возможность применения данного алгоритма для решения целого класса задач, отвечающих общей постановки задачи. Так как алгоритм, показанный на рисунке, позволяет правильно подсчитать сумму не только чисел 2 и 3, но любой другой пары целых чисел, он обладает свойством массовости. Для того чтобы алгоритм обладал свойством массивности, следует составлять алгоритм, используя обозначения величин и избегая конкректных значений.

*Правильность алгоритма*, под которой понимается способность алгоритма давать правильные результаты решения поставленных задач. Представленный в примере алгоритм обладает свойством правильности, так как в нём использована правильная формула сложения целых чисел, и для любой пары целых чисел результат выполнения алгоритма будет равен их сумме.

Компьютерная программа – это план будущих работ, составленный в расчёте на его выполнение компьютером.

Чтобы компьютер смог выполнить программу, она должна быть записана в специальной форме, доступной компьютеру; должна быть записана в соответствии со специальным набором правил.

*Набор записи компьютерной программы называется “алгоритмическим языком”.*

***1.2.1 Линейные алгоритмы.***

Вычислительный процесс называется линейным (не разветвляющимся), если направление его продолжения на любом этапе вычислений является единственным. Алгоритм линейного вычислительного процесса описывает действия, последовательность выполнения которых не зависит от исходных данных и результатов промежуточных вычислений, т.е является постоянной. Этот процесс является наиболее простым видом вычислений. Линейный процесс (как и другой вычислительный процесс) можно представить в виде следующих этапов: первый – задание исходных данных; второй реализация вычислений; третий – вывод результатов счёта и поясняющей информации. Этапы отображаются на блок-схеме, а затем реализуются в ПЭВМ в указанной последовательности.

Алгоритм деления отрезка АВ пополам:

1) поставить ножку циркуля в точку А

2) установить раствор циркуля равным длине отрезка АВ

3) провести окружность

4) поставить ножку циркуля в точку В

5) провести окружность

6) через точки пересечения окружностей провести прямую

7) отметить точку пересечения этой прямой с отрезком АВ

Каждое указание алгоритма предписывает исполнителю выполнить одно конкретное значения действий. Исполнитель не может перейти следующей операции, не завершив полностью предыдущую. Предписания алгоритма надо выполнять последовательно одно за другим, с соответствии с порядком их записи. Следование всем предписаниям гарантирует правильное решение задачи. Данный алгоритм совершенно ясен исполнителю

*Блок-схема* – алгоритм выраженный с помощью логических блоков. Блок – схема служит для того, чтобы наглядней представлять те или иные формы организаций действий. Каждое действие алгоритма, кроме проверки условия, будем помещать в прямоугольник, а вопрос о том, выполняется ли некоторое условие, - в ромб. Еще существуют: параллелограмм, овал, оборванный листок,

– это блок ввода данных с клавиатуры.

– в этом блок указывается начало или конец алгоритма

- это блок вывода данных на печать.

- в этом блоке помещаются действия алгоритма.

- блок в котором помещаются условия.

Вот так выглядит блок-схема линейной функции.

Подойти к ларьку

Протянуть деньги

Попросить газету.

Взять газету и сдачу

Отойти от ларька

***1.2.2 Алгоритм с ветвлением.***

В рассмотренных до сих пор алгоритмах и программах все команды выполнялись последовательно одна за другой в том порядке, в каком они были записаны. Однако таким образом может быть построен алгоритм для решения далеко не всякой задачи. В практике известны задачи, дальнейший ход решения которых зависит от выполнения каких либо условий.

Рассмотрим простой пример задачи из курса алгебры. Требуется построить алгоритм вычисления значения функции у = I x I. Она задаётся соотношением

Х при х >=0,

У = ⎨

-x при

При решении этой задачи требуется выполнить следующие условия.

1. Проверить больше или равен нулю х
2. Если х больше или равен 0, то присвоить у значение х (у:=x),

Если х меньше 0, то присвоить у значение –х(у:=-х).

Коротко алгоритм решения этой задачи может быть записан так:

Если x>=0,

ТО y:=x,

Иначе y=-x

Команды, с помощью которых записывается алгоритм подобного типа (разветвляющие алгоритмы), называются *командами ветвления.*

*Ветвление* - это такая форма организаций действий, при которой в зависимости от выполнения или невыполнения некоторого условия совершатся либо одна, либо другая последовательность действий.

Блок схемы на рисунках а, б, в, изображают соответственно последовательное выполнение действий (линейный алгоритм), ветвление в полной и неполной формах.

А) б)

В)

На рисунке изображена блок – схема алгоритма покупки билетов.

Подойти к кассе

Билеты на 12.00 имеются? имеются

Да Нет

Протянуть кассиру деньги

Назвать сеанс и количество билетов

Получить билеты

Отойти от кассы

Данная блок схем отображает ветвление в краткой форме, когда дей ствия и дут по одной ветке.

Но существует еще ветвление с полной формой ,когда действия идут по обоим веткам

Уменьшить скорость

Ремонт участка закончен

Проехать по 5 км отремонтированной шоссе

Проехать 10 км в объезд

пропро

Остановиться у АЗС

Для отработки составления блок схем с ветвлением я рассмотрел несколько алгоритмов и составил к ним блок схемы:

А) Присвоить х значение суммы углов А и С Четырёхугольника ABCD.

Присвоить y значение суммы углов B и D четырёхугольника ABCD.

Если х=у, то:

Построить серединный перпендикуляр к отрезку AB.

Построить серединный перпендикуляр к отрезку BC.

Найти пересечение построенных перпендикуляров.

Иначе:

Сообщить “Построение невозможно”.

Конец ветвления.

Б) Присвоить х значение суммы сторон AB и CD четырёхугольника ABCD.

Присвоить у значение суммы сторон BC и AD четырёхугольника ABCD.

Если х=у, то:

Построить биссектрису угла А.

Построить биссектрису угла В.

Найти пересечение построенных биссектрис.

Иначе:

Сообщить “Построение невозможно”.

Конец ветвления.

А)

# ABCD

Присвоить х значение суммы углов А и С Четырёхугольника ABCD.

Присвоить y значение суммы углов B и D четырёхугольника ABCD.

# Если Х=У

Построить серединный перпендикуляр к отрезку BC.

Найти пересечение построенных перпендикуляров.

Вывести сообщение “Построение невозмоджно”

Построить серединный перпендикуляр к отрезку BC.

Найти пересечение построенных перпендикуляров.

Б)

ABCD

Присвоить х значение суммы сторон AB и CD четырёхугольника ABCD.

Присвоить у значение суммы сторон BC и AD четырёхугольника ABCD.

X=Y

Построить биссектрису угла А.

Построить биссектрису угла В.

Найти пересечение построенных биссектрис.

Вывести сообщение “построение невозможно”

Конец

**II. Ветвление в Паскале**

**2.1 Языки программирования.**

Чтобы компьютер выполнил решение какой – либо задачи, ему необходимо получить от человека инструкции, как её решать. Набор таких инструментов для компьютера, направленный на решение конкретной задачи, называемой *компьютерной программой.*

Современные компьютеры не настолько совершены, чтобы понимать программы, записанные на каком – либо употребляемой человеком языке – русском, английском, японским… Команды, предназначенные для ЭВМ, необходимо записать в понятной её форме. С этой целью применяются языки программирования – искусственные языки, алфавит, словарный запас и структура которых удобны человеку и понятнее компьютеру.

В самом общем смысле языком программирования называется фиксированная система обозначений и правил для описания алгоритмов

и структур данных. Языки программирования имеют как бы два лица. Одно из них обращено к человеку, использующему язык для записи своих программ, а другое адресовано ЭВМ, которая должна понимать команды.

Исходя из этого все языки программирования делятся на языки низкого, высокого и сверхвысокого уровня.

Языки низкого уровня – это средство записи инструкций компьютеру простыми приказами – командами на аппаратном уровне. Такой язык отражает структуру данного класса ЭВМ и поэтому иногда называется Машинно – ориентированным языком. Пользуясь системой команд, понятной компьютеру, можно описать алгоритм любой сложности. Правда, такая запись для сложных задач будет на столько громоздкой, что у человека будет мало шансов сделать её безошибочной, так как этот язы мало приспособлен для использования человеком, ведь запись программы на этом языке представляет собой последовательность нулей и единиц.

Существенной особенностью языков программирования низкого уровня жесткая ориентация на определённый тип аппаратуры (систему команд процессора). В стремлении приспособить язык программирования низкого уровня к человеку разработан язык символического кодирования (автокод или язык ассемблера), структура команд которого определяется форматами команд и данными Машиного языка. Программа на этом языке ближе к человеку, потому что операторы этого языка – те же команды, но они имеют мнемонические названия, а в качестве операндов используются не конкретные адреса в оперативной памяти, а их символические имена.

Более многочисленную группу составляют языки программирования высокого уровня, средства которых допускают описание задачи в наглядном, легко воспринимаемом виде. Отличной особенностью этих языков является их ориентация не на систему команд той или иной ЭВМ, а на систему операторов, характерных для записи определённого класса алгоритмов. К языкам программирования этого типа относятся: Бейсик, Фортран, Алгон, Паскаль, Си. Программа на языках высокого уровня записывается системой обозначений, близкой человеку (например, фиксированным набором слов английского языка, имеющих строго определённое назначение). Программу на языке высокого уровня проще понять и значительно легче отладить.

К языкам программирования сверхвысокого уровня можно отнести Алгон-68, при разработке которого сделана попытка формализовать описание языка, приведшая к появлению абстрактной и конкретных программ. Абстрактная программа создаётся программистом, конкретная выводится из первой. Предполагается, что при таком подходе принципиально невозможно породить неверную синтаксически ( а в идеале семантически) конкретную программу. Язык APL относят к языкам свервысокого уровня за счёт введения сверхмощных операций и операторов. Запись программ на таком языке получается компактной.

Все вышеперечисленные языки – вычислительные. Более молодые – декларативные (непроцедурные) языки, отличительная черта которых – задание связей и отношений между объектами и величинами и отсутствие определения последовательности выполнений действий. Такие языки сыграли важную роль в программировании, так как они дали толчок к разработке специализированных языков искусственного интеллекта и языков знаний.

**2.2 Язык программирования Паскаль.**

Язык программирования Паскаль (назван в честь выдающегося французского математика и философа Блеза Паскаля (1623 – 1662)), разработан в 1967 – 1971гг. Никлаусом Виртом, профессором, директором института информатики Швейцарской высшей политехнической школы. Язык Паскаль, созданный первоначально для обучения программированию как систематической дисциплине, скоро стал широко использован для разработки программных средств в профессиональном программировании.

Широкой популярностью Паскаля среди программистов способствовали следующие причины:

- Благодаря своей компактности, удачному первоначальному описанию Паскаль оказался достаточно лёгким для обучения.

- Язык программирования Паскаль . Отображает фундаментальные и наиболее важные концепции (идеи) алгоритмов в очевидной и легко воспринимаемой форме, что предостовляет программисту средства, помогающие проектировать программы.

- Язык Паскаль позволяет чётко реализовать идеи структурного программирования и структурной организации данных.

- Язык Паскаль сыграл большую роль в развитии методов аналитического доказательства правильности программ и позволил реально перейти от методов отладки программ к системам автоматической проверки и правильности программ.

- Применения языка Паскаль значительно подняло “планку” надёжности разрабатываемых программ за счёт требований Паскаля к описанию используемых в программе переменных при компиляции без её выполнения.

- Использование в Паскале простых и гибких структур управления: ветвлений, циклов.

**2.3 Условный оператор в Паскале.**

Условные операторы предназначены для выбора к исполнению одного из возможных действий ( операторов) в зависимости от некоторого условия (при этом одно из действий может быть пустым, т.е. отсутствовать) В качестве условий выбора используется значение логического выражения. В Турбо Паскале имеются два вида условных операторов: if иcase.

Оператор условиа if. Оператор условия if является одним из самых популярных средств, имеющих естественный порядок выполнения операторов программы. Синтаксическая диаграмма оператора if выглядит таким образом:

Выражение

Оператор

оператор

Операторы условия if выполняются следующим образом. Сначала выражается выражение, записанное в условии. В результате его вычисления получается значение булевского типа. В первом случае, если значение выражения есть true (истина), выполняется <оператор 1>, указанный после слова Then (то). Если результат вычисления выражений в условии естьFalse (ложь), то выполняется <оператор 2>. Во втором – если результат выражения true, то выполняется <оператор>, если false – оператор, следующий сразу за оператором if. Операторы if могут быть вложенными.

Read (Ch);

If Ch=’N’ then parol:= True

Else Parol := False;

Read (x);

If Parol = True then

If x = 100 then Writeln (‘пароль и код правельны’)

Else begin

Writeln (‘Ошибка в коде’);

End;

В данном примере с клавиатуры считывается значение переменной символьного типа Ch. Затем проверяется условие Ch=’N’. Если оно выполняется, то переменной Parol булевского типа присваеватся значение True, если условие не выполняется, False. Затем с клавиатуры считывается значение кода Х. Далее оператор проверяет условие Parol = True. Если оно имеет значение True, то выполняется проверка введённого пароля оператором if X=100. Если условие Х=100 имеет значение True, то выводится сообщение “Пароль и код правельны”, и управление в программе передаётся на оператор, следующий за словом end, если оно имеет значение False, выполняется соствной оператор, стоящий после слова else, который выводит на экран видеомонитора сообщение “ошибка в коде”, и вызывает стандартную процедуру Halt(1) для остановки программы.

* + 1. **Составной оператор (оператор варианта).**

Условный оператор позволяет осуществить ветвление программы только по двум направлениям, одно из которых соответствует выполнению проверяемого условия. Если для переменной необходимо осуществить ряд действий, зависящих от других условий, то надо записывать либо вложенные условные операторы, либо несколько операторов подряд. Для такой операции удобно использовать оператор варианта.

Вход в структуру содержит вычисление или ранее полученное значение переменной (индекса варианта). Это значение может совпасть с меткой, стоящей перед оператором на одной из ветвей переключаются. В таком случае выполняется оператор, помеченный этой меткой, и происходит выход из структуры. Оператор бывает простым или составным, ограниченым операторными скобками *begin … end***;** Если значение индекса варианта не совпало ни с одной из меток, то выполняется оператор с номером n + 1 из строки *else***.** Если оператор варианта содержит строку *else* , то это - полная форма оператора, если такой строки нет, то используется сокращённая форма оператора варианта.

*Метки* оператора варианта могут быть константами любого типа. Их тип должен совпадать с переменной индекса варианта. Возможно, чтобы индекс варианта должен был как именем переменной, так и выражением соответствующего типа.

В рассмотренных до сих пор алгоритмах и программах все команды (операторы) выполнялись последовательно одна за другой в том порядке, в котором они были записаны. Однако таким образом может быть построен алгоритм для решения далеко не всякой задачи.

В практике хорошо известны задачи, дальнейший ход решения которых зависит от выполнения какого либо ***условия***.

Рассмотрим простой пример задачи из курса алгебры. Требуется построить алгоритм вычисления значений функции у = |х|. Она задаётся соотношением:

Y =

На языке Паскаль оператор варианта имеет вид:

**Case** индекс варианта **of**

# Метка 1: **оператор 1**;

Метка 2: **оператор 2**;

…

Метка n: **оператор n;**

**Else оператор n + 1**

**End;**

# Пример:

Составить программу-решатель.

Дано: молярная масса вещества М, плотность данного вещества Р, масса R или объём V. Требуется найти число молекул К: 1) в единице массы вещества; 2) в теле заданной массой: 3) в единице объёма вещества; 4) в теле заданным объёмом. Для решения задачм воспользуемся формулой:

K = (NА/M)R,

Где NA = 6,022 \* 10²³г/моль – число Авогадро.

На основании этой формулы получаем расчётные формулы для программы:

1)K=NA/M; 2)K=NAR; 3)K=NAPV/M 4)K=NAP/M.

Программа имеет вид:

Program E20;

Const NA = 6.022 E20;

Var N: integer; M, R, P, V, K: real;

Begin

Writeln (‘зная постоянную Авогадро, Плотность P данного вещества’);

Writeln (‘и его молярную массу М, можно найти число молекул в’);

Writeln (‘1. в единице массы вещества’);

Writeln (‘2. в теле массой R’);

Writeln (‘3. в единице объёма’);

Writeln (‘4. в теле объёмом V’);

Write (‘введите номер решающей задачи’);

Readln (N);

Write (‘введите исходные данные: М =’);

Readln (M);

Case N of

1: K: = NA / M;

2: begin

Write (‘R = ‘);

Readln (R);

K: = NA \*R/M;

End;

3: begin

write (‘плотность вещества Р = ’);

Readln (P);

Write (‘V = ‘);

Readln (V);

K: = NA \***P**\*V/M;

End;

4: begin write (‘плотность вещества P = ’);

Readln (P);

K: = NA \*P/V

End;

End;

Write (‘число молекул K = ’, K’);

End.

**2.3.2 Вложенные условия.**

Существуют задачи в которых по одной из ветки ветвления нужно рассмотреть ещё одно условие.

Рассмотрим задачу: сравнить 3 целых числа и вывести на печать максимальное.

Program chisla;

Var a, b, c: integer;

Begin

Writeln (‘Введите три числа’);

Read (a, b, c);

If a<b then

If a>c then writeln (‘ максимально число ’, a)

Else if b>c then writeln (b максимально число’, b)

Else writeln (‘c максимальное число’, c);

End.

Вот блок – схема данной задачи:

a>b

b>c

a>c

B

C

A

ABC

**2.3.3 Логические выражения.**

Алгоритм решения квадратного уравнения содержит проверку условия D<0. Два значения, D и 0, связаны отношением < - меньше. Если условие выполняется то говорят что соответствующее выражение истинно, если не выполняется то ложь.

Речь идёт о логическом выражении. Примером логических выражений могут служить математические отношения связанные знаками >, <, =, логические выражения имеют два значения True( истина) и False (ложно). Логические выражения такие могут иметь логические операции: отрицания (NOT), конъюнкций (OR) и дизъюнкции (AND).

Логические операции имеют следующую таблицу истинности:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | Not (A) | (A) AND (B) | (A) OR (B) |
| True | True | False | True | True |
| True | False | False | False | True |
| False | True | True | False | True |
| False | False | True | False | False |

Пусть требуется определить, принадлежит ли точка с координатой x отрезку [a; b]. Если записать это условие двойным неравенством, то читать его надо так: x меньше либо равен b и больше либо равен a (a<x<b). Отношение “меньше или равно” в Паскале записывается двумя знаками. Аналогично записывается и “больше либо равно”. Однако в Паскле нельзя записывать двойное неравенство. Используя логическую операцию and (и), запишем:

(x>=a) and (x<= b).

Отношения, между которыми стоит логическая операция, заключаются в круглые скобки.

Имеется прямоугольное отверстие со сторонами a и b и кирпичь с рёбрами x, y z. Требуется составить условие прохождения кирпича в отверстие.

Кирпич пройдёт в прямоугольное отверстие, если выполняется сложное условие:

(a>=x) and (b>=y) or

(a>=y) and (b>=x) or

(a>=x) and (b>=z) or

(a>=z) and (b>=x) or

(a>=y) and (b>=z) or

(a>=z) and (b>=y)

Для трёх граней шесть условий получается потому, что можно каждую грань повернуть на 90 и проверить для каждой грани два случая.

Определить принадлёжность точки фигуре. Пусть фигура задана ограничивающими её прямыми.

Для каждой прямой определим полу плоскость, в которой находится фигура- треугольник АВС. Полуплоскость задаётся неравенством.

Полуплоскость, находящаяся выше от оси х определяется неравенством y>0.

Полуплоскость, находящаяся справа от прямой, соединяющей точки (-1,0) и (0,2), задаётся неравенством y-2x-2<0.

Полуплоскость, находящаяся слева от прямой, соединяющей точки (1,0) и (0,2), задаётся неравенством y+2x-2<0.

Условие принадлежности точки (x,y) фигуре :

(y>0) and (y-2\*x-2<0) and (y+2\*x-2<0).

Приведём пример программы определяющая существования треугольника со сторонами a, b и c. Условие существования треугольника известно из геометрии: сумма двух любых сторон должна быть больше третьей. Следовательно, для всех сторон условие “сумма двух больше третьей” должно выполняться.

Program e4;

Var a, b, c: real;

Begin

Writeln (‘дите длины трёх сторон треугольника’);

Readln (a, b, С);

Write (‘треугольник со сторонами ’, a, b, c);

If (a+b>c) and (b+c>a) and (a+c>b)

Then writeln (‘существуют’)

Else writeln (‘не существуюет’);

Readln

End.

**Ш . Технологический пакет к урокам информатики по теме “Ветвление в Паскале”**

**Задачи с ветвлением и их решение.**

Program vetvlenie;

Var t: integer;

Begin

Writeln ('Введите температуру сегодня');

Readln (t);

If t<=-30

Then

Writeln (‘Не идти в школу ')

Else writeln (‘Идти в школу');

End.

Программа показывает идти ученику сегодня в школу или нет. С клавиатуры он вводит сегодняшнюю температуру, и если она меньше -30, то не идти в школу, а если больше то выводится сообщение о том чтобы идти в школу. Теперь попробуйте написать программу:

*Сделать так чтобы программа выводила сообщение о том у кого больше денег и у кого меньше.*

*Переделать программу так чтобы вместо денег выводилось сообщение о том кто больше попрыгал. Значения так же вводятся с клавиатуры.*

Рассмотри задачу: Для модельного агентства выбираются девушки с объёмами 90 (грудь), 60 (талия), 90 (бёдра). Составить программу для отбора девушек по этим параметрам.

ABC

A= 90

B= 60

C= 90

Вы нам не подходите

Вы нам подходите.

Конец

**Вопросы и задачи по всей теме.**

Составьте программу, которая из двух вводимых вами целых чисел печатает заключение о том, какое число больше

Составьте программу, проверяющую, верно ли утверждение, что введённое вами число является чётным.

Составить программу, определяющую, входит ли введенная вами цифра в десятичную запись введенного вами трёхзначного числа, и печатуящая сообщение о том, входит ли эта цифра в запись числа или нет.

Напишите программы на Паскале для решения следующих задач.

Даны три числа a, b, c. Выясните верно ли a<b<c. Ответ получите в текстовой форме: верно или не верно.

Даны положительные числа a, b, c, x. Выясните, пройдёт ли кирпич с рёбрами a, b, c в квадратное отверстие со стороной x.

Даны три действительных числа. Выберете те из них, которые принадлежат отрезку [1;3].

Присвойте величине а значение наибольшего из трёх заданных чисел.

Даны два числа. Выведите первое из них, если оно больше второго, и оба числа, если это не так.

Проверьте, если среди трёх заданных чисел равные.

Даны три действительных числа. Найдите наибольшее значение их суммы и произведения.

Напишите программу – модель анализа пожарного датчика в помещении, которая выводит сообщение «пожарная ситуация», если температура (в нашей модели она будет вводиться с клавиатуры) в комнате превысила 600 .

Составьте программу, которая из двух вводимых вами чисел, печатает заключение о том, какое число больше.

Составьте программу, проверяющую, верно ли утверждение, что введённое целое число делится без остатка на 3.

Напишите программу, которая анализирует человека по возрасту и относит его к одному из четырёх групп: дошкольник, ученик, рабочий, пенсионер. Возраст вводится с клавиатуры.

Составьте программу, определяющую, входит ли введённая вами цифра в десятичную запись введённого вами трёхзначного числа, и печатающую сообщение о том, входит ли эта цифра в запись числа или нет.

Составьте программу, которая по трём введённым вами числам определит,

могут ли эти числа быть длинами сторон треугольника, если да, то какой получился треугольник с данными длинами сторон (прямоугольный, остроугольный, тупоугольный).

Составьте программу, определяющую, лежит ли точка с указанными координатами X, Y на окружности радиуса R с центром начала координат.

Составьте программу нахождения корня уравнения ax+b=c, где a, b, c – данные числа.

Даны три действительных числа. Возвести в квадрат те из них, значения которых не отрицательны.

Даны три действительных числа. Выбрать из них те, которые принадлежат отрезку {-5;3}.

Известны результаты соревнования трёх лыжник4ов. Выбрать результат победителя и занявшего третье место.

Дано натуральное четырёхзначное число. Верно ли, что все четыре цифры различны?

Дано натуральное число n(n<100), определяющее возраст человека (в годах). Дать для этого числа наименование «год», «года» или «лет»: например, 21 год, 44 года, 65 лет и т.д.

**Тест.**

Что такое алгоритм?

А) последовательность выполнения действий.

Б) составление программы.

В) Не знаю.

Что такое Блок-схема?

А) Рисунки квадратиков.

Б) графическое изображение алгоритма.

В) Не имеющие смысла геометрические фигуры.

Для чего нужно ветвление в паскале?

А) Для решения более сложных задач.

Б) Для того чтобы больше подумать над решением задачи.

В) я не знаю.

Какое из этих языков программирования не относится к высшему уровню?

А) Паскаль

Б) Бейсик

В) Машинно – ориентированный

Где чаще всего используется язык программирования Паскаль?

А) Дома

Б) В хакерских группировках

В) В школе и других учебных заведениях.

Каких алгоритмов не существует?

А) Корявых

Б) линейных

В) С ветвлением.

Каких операторов не существует?

А) Условный.

Б) Заумный.

В) Составной.

Почему язык Паскаль такой популярный?

А) Он простой в обучении и изучается в школах

Б) Очень сложный Язык программирования и это мне нравится.

В) Я не знаю.

С какой командой начинается программа?

А) начало

Б) var

В) begin

Как выглядит оператор присваивания?

А) :=

Б) :()

В) =;

Что указывается в операторах?

А) условие

Б) переменные

В) начало и конец.

**Заключение.**

Цель моей работы достигнута: я рассмотрел различные виды алгоритмических конструкций, изучил конструкцию ветвления, научился решать задачи с ветвлением, подобрал и составил сам задачи, тесты и др. задания.

В дальнейшем я бы хотел продолжить свою работу, рассмотрев в Паскале другие конструкции. Так же мне хотелось бы узнать, как описывается конструкция ветвления в других языках программирования.(C++, Delphi)

126 (6-16)

**Список литературы**

Введение в програмирование: Учеб. Пособие для уч. Сред.и ст. шк. Возраста / Авт.-сост. В.А. Гольденберг.- Мн.: ООО”Харвест”, 1997.- 528 с. (библиотека школьника).

Бондарев В.М., Гублинский В.И., Качко Е.Г..

Основа програмирования/ худож. – оформитель С.А. Пяткова. – Харьков: Фолио; Ростов Н/Д: Феникс, 1997. –368с.

Турбо Паскаль 7.0 – К.: торгово – издательское бюро BHV, 1996 – 448с.: ил. Т. Рюттяна.

Игошев А.Д. Матинин Е.Г. Express Pascal. Учебное пособие для средних уч. Заведений.

Кузницов А.А. Патапова Н.В. Основы информатики. 8-9 кл.: Учеб. Для общеобразовательных учеб. Заведений 2001 176с.:ил.