**Практическая работа 6**

**Типы данных**

**Ответьте на вопросы:**

1. Что такое запись? Из чего она состоит. Приведите свой пример.

**Записи в Паскале – структурированный комбинированный тип данных.**Она состоит из элементов – полей, каждый из которых обладает определенным типом данных. Пример: запись о студенте ВятГУ, в которой храниться ФИО, ДР, № зачетной книжки и студенческого билета и т.д.

1. Как объявляются и создаются записи в Паскале? Как обращаться к полям записи? Приведите примеры.

Записи в паскале создаются при помощи оператора record. Для создания записи необходимо записать ее имя, затем прописать record, и в операторных скобках указать все необходимые поля и их типы. Для обращения к полям записи необходимо прописать их идентификатор и через точку указать необходимое поле. Например:

Student.Name := ‘Mikhail’;

Student.Year := 18;

1. Зачем используются записи в виде двумерной таблицы? Приведите пример использования.

Для использования записей в виде двумерной таблицы необходимо использовать одномерный массив, состоящий из записей. Такой тип данных позволяет хранить большое количество различных записей и удобно с ними работать: печатать, записывать в файл, добавлять новые элементы таблицы. Пример:

Type food = record

…

End;

Var

Food\_array ; array[1..100] of food;

Begin

For var i:=1 to 100 do

… (writeln, read, := )

End;

1. Как использовать конструкции with при работе с записями?

Конструкцию with необходимо использовать для более удобного присваивания значений полям записи. Пример:

With Stident do

Begin

Name := …;

Year := …;

…

End;

1. Как происходит считывание/запись записей в файл?

Для того, чтобы работать с записями при помощи файла, необходимо использовать типизированный файл, с типом этой записи. После создания такого файла в него можно свободно заносить и извлекать записи, при помощи их идентификаторов или двумерной таблицы. Пример:

Var record\_file: file of student;

1. Что такое множество в Паскале? Как оно задается?

Множество – набор элементов одного типа. Множество задается при помощи конструкции set of (тип). Пример: var num\_set : set of integer := [0..9];

1. Какие действия можно выполнять с множествами? Приведите примеры.

7.1 Проверять наличие какого либо элемента в множества при помощи оператора in;

7.2 Выполнять объединение множеств при помощи оператора +;

7.3 Вычислять объекты, находящиеся в 2 множествах одновременно при помощи оператора \*;

7.4 Вычислять объекты, которые есть в первом множестве, но нет во вором при помощи оператора -;

1. Для чего используется операция IN при работе с множествами? Приведите примеры.

Операция in используется для проверки наличия элемента в множестве;

…

Read(a);

If a in num\_set then writeln(‘Это цифра’);

…

1. Опишите динамические данные, в чем их особенность?

Динамические данные, это данные, которые хранятся в динамической области оперативной памяти, которая выделяется под них непосредственно в процессе работы программы.

1. Что такое указатель? Как он объявляется? Как присваиваются значения? Приведите примеры.

Указатель – это переменная, которая ссылается на определенную область динамической памяти. Для объявления указателя необходимо перед его типом поставить знак ^. Для присвоения указателю какого либо значения необходимо либо использовать адрес уже существующей ячейки (через @), либо использовать оператор new (указатель) и занести нужное значение. Пример:

Var a: ^integer; b: pointer;

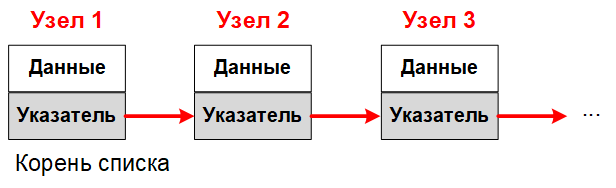
1. Что такое список? Какие виды списков бывают?

Список – это динамическая структура данных, которая состоит из нескольких элементов. Каждый такой элемент состоит из полей с данными и поля с указателем на следующий элемент. Таким образом, список можно представить в виде последовательности его элементов, связанных между собой.

Выделяют 2 основных видов связанных списков – односвязный и двусвязный списки.

1. Что такое односвязный список? Как он выглядит? Как происходит объявление типа данных?

Односвязный список представляет собой набор записей, хранящихся в динамической памяти, каждая из которых имеет ссылку только на следующую запись. Также для каждого односвязного списка должен храниться указатель на первый элемент списка – голова списка, с которой можно пройтись по всему списку. Выглядит этот список следующим образом:



(голова)

1. Опишите алгоритм работы с алфавитно-частотным словарем. Опишите используемые подпрограммы.

Алгоритм работы с алфавитно-частотным словарем:

* Проверить, осталось ли слово в файле;
* Если слово есть, извлечь его из файла;
* Проверяем, есть ли в списке хоть один элемент, т.е. его голова, и больше ли ее слово, слова нашего элемента. Если головы нет, либо она больше нашего элемента, необходимо ставить элемент в начало списка;
* В противном случае необходимо начать продвижение по списку, предварительно присвоив указателю продвижения указатель головы.
* Если при продвижении находим элемент со словом, равным нашему, увеличиваем счетчик этого элемента и заканчиваем продвижение.
* Если находим элемент со словом меньше нашего необходимо осуществить ряд проверок. Если этот элемент не указывает на следующий, т.е. мы достигли конца списка, нужно вставить новый элемент за ним и закончить продвижение. Если слово следующего за ним элемента больше нашего, т.е. необходимо вставить новый элемент перед ним, осуществляем процедуру добавления элемента после текущего проверяемого (примечание к следующему элементу можно обратиться по ссылке текущего, не переходя к следующей итерации, так мы на 100% можем быть уверены, что ситуация, когда текущий элемент будет больше нашего и нам необходимо вставить элемент перед ним невозможна). В противных случаях необходимо изменить текущий элемент на следующий и продолжить движение по списку.

Программа изъятия слова из файла:

- запускаем цикл, в котором будем проходить по файлу и считывать его элементы, пока не встретим букву.

- запускаем цикл, в котором будем проходить по файлу, пока не встретим любой знак. В процессе работы цикла необходимо считывать элементы файла и формировать слово-результат;

- в конце переводим все буквы в нижний или в верхний регистр для корректного формирования словаря.

Программа добавления слова в начало:

- Присваиваем полю next нового элемента указатель головы списка Head;

- Голове списка Head присваиваем указатель нового элемента.

Программа добавления элемента после заданного:

- Присваиваем полю next нового элемента поле next того элемента, после которого необходимо его добавить;

- Присваиваем полю next элемента, после которого необходимо добавить новый элемент, указатель на новый элемент;

1. Что такое двусвязные списки? В чем особенность работы с двусвязными списками?

Двусвязные списки – это разновидность связанных списков, в которых элементы ссылаются не только на следующий элемент, но и на предыдущий.

1. **Измените алгоритмы для работы с алфавитно-частотным словарем, используя двусвязные списки.**
2. **Выведите список слов из словаря в порядке убывания частоты, то есть, сначала те слова, которые встречаются чаще всего.**
3. Что такое стек? Приведите пример.

Стек – это динамическая структура данных, которая представляет собой последовательность элементов, каждый из которых ссылается на предыдущий. Для стека существует правило LIFO, первый вошел, первый вышел, т.е. добавлять и удалять элементы можно только из конца стека. Примеры – рекурсивный стек, стопка бумаги.

1. Поясните основной принцип стека.

LIFO

1. Опишите основные операции со стеком. Опишите используемые подпрограммы.

Основные операции над стеком – извлечение элемента, добавление элемента, проверка пустоты стека.

Для добавления элемента, необходимо указателю на добавляемую запись в поле, указывающий на следующий элемент присвоить значение указателя на голову стека, а голове стека присвоить область памяти, в которой находится добавляемый элемент стека;

Для удаления элемента необходимо вернуть значение, на которое указывает указатель head, а затем переназначить указатель head, на тот элемент, на который указывает текущее значение данного указателя.

Для проверки пустоты стека необходимо проверить указатель head, если он равен nil, значит стек не содержит элементов.

1. Что такое очередь? Приведите пример.

**Очередь** — динамическая структура данных, у которой в каждый момент времени доступны только два элемента: первый и последний. Добавление элементов возможно только с одного конца (конца очереди), а удаление элементов – только с другого конца (начала очереди). Примеры: очередь в магазине, на сдачу лабораторной работы.

21.Поясните основной принцип очереди.

Добавление элементов возможно с одного конца, а удаление с начала FIFO;

1. Опишите основные операции с очередями.

Добавление элемента в очередь – для добавления элемента в очередь необходимо в поле адреса этого элемента указать значение указателя на хвост очереди, а указатель хвоста необходимо перенаправить на добавляемый элемент.

Для удаления элемента из очереди необходимо возвратить значение из указателя головы очереди head, а указателю головы очереди необходимо определить значение следующего за ним элемента.

1. Что такое дек? Приведите пример.

Дек – это очередь, в которой добавлять и удалять элементы можно с обоих концов.

1. Опишите подпрограммы из задания с деком.

Они не были обнаружены.

1. Что такое дерево? Перечислите основные элементы дерева.

Дерево – это частный случай графа – структура данных, которая состоит из узлов – элементов, и ребер – связей между ними. В отличие от графов деревья не могут содержать циклов, а также более одного внешнего соседа.

В дереве выделяют 3 основных элемента – корневой элемент - первый элемент дерева, который не имеет родителя;

Родительский элемент – элемент, который имеет соседние элементы и сам является дочерним какому либо элементу;

Листовой элемент – элемент, не имеющий дочерних;

1. Что такое двоичное дерево?

Двоичное дерево – это дерево, каждый узел которого должен иметь не более двух дочерних узлов, причем все узлы правее данного, должны быть больше этого элемента, а все узлы левее данного должны быть меньше;

1. Опишите алгоритм поиска по дереву.

Поиск в глубину:

* Начинаем с корневого узла.
* Если он равен искомому элементу, то прекращаем поиск.
* Если он не равен искомому элементу, то необходимо проверить все его имеющиеся ветсви рекурсивно

1. Опишите алгоритм поиска в бинарном дереве.

* Начинаем проверку с корневого элемента;
* Проверяем, если элемент равен нашему, поиск завершен;
* Если больше нашего, переходим в правую ветвь;
* Если меньше нашего, переходим в левую ветвь;

\*Если ветви при переходе не существует, т.е. указатель равен nil, соответственно элемента в бинарном дереве нету.

1. Улыбнитесь😊