Уважаемые студенты групп!

Вашему вниманию представлена лекция на тему «ПОСТРОЕНИЕ ПРОГРАММ ДЛЯ СОЗДАНИЯ ФАЙЛОВ ПРЯМОГО И ПОСЛЕДОВАТЕЛЬНОГО ДОСТУПА»

Задание

1. Прочитать внимательно лекцию.
2. Законспектировать лекцию в рабочую тетрадь не менее 3-5 страницы рукописного текста. В конспекте лекции обязательно должно быть приведены примеры.
3. Решить приведенные в лекции в контрольных вопросах задачи.
4. Дата предоставления фотоотчета лекции до 03.05.2023.

С уважением Ганзенко Ирина Владимировна

!!! Если возникнут вопросы обращаться по телефону 0721134803 (вацап), +79591134803 (телеграмм)
disobuch.ganzenko2020@mail.ru

ПОСТРОЕНИЕ ГРАФИКОВ ФУНКЦИЙ

1 Теоретическое положение

Существенной особенностью всех рассмотренных до сих пор значений производных типов является наличие в них конечного, наперед заданного числа компонент. Так, в значении многомерного массива это число можно определить, зная количество компонент по каждому измерению, а в значении записи это число определяется количеством и типом полей. Таким образом, заранее, еще до выполнения программы, по этому описанию можно выделить необходимый объем памяти машины для хранения значений переменных этих типов. Но существует определенный класс задач и определенные ситуации, когда количество компонент (пусть даже одного и того же из известных уже типов) заранее определить невозможно, оно выясняется только в процессе решения задачи. Поэтому возникает необходимость в специальном типе значений, которые представляют собой произвольные последовательности элементов одного и того же типа, причем длина этих последовательностей заранее не определяется, а конкретизируется в процессе выполнения программы. Этот тип значений получил название **файлового типа**. Условно **файл в Паскале** можно изобразить как некоторую ленту, у которой есть начало, а конец не фиксируется. Элементы файла записываются на эту ленту последовательно друг за другом:

F	F1	F2	F3	F4	. . . .

где F – имя файла, а F1, F2, F3, F4 – его элементы. Файл во многом напоминает магнитную ленту, начало которой заполнено записями, а конец пока свободен. В программировании существует несколько разновидностей файлов,

отличающихся методом доступа к его компонентам:**файлы последовательного доступа** и **файлы произвольного доступа**.

Простейший метод доступа состоит в том, что по файлу можно двигаться только последовательно, начиная с первого его элемента, и, кроме этого, всегда существует возможность начать просмотр файла с его начала. Таким образом, чтобы добраться до пятого элемента файла, необходимо, начав с первого элемента, пройти через предыдущие четыре. Такие файлы называют **файлами последовательного доступа**. У последовательного файла доступен всегда лишь очередной элемент. Если в процессе решения задачи необходим какой-либо из предыдущих элементов, то необходимо вернуться в начало файла и последовательно пройти все его элементы до нужного.

Файлы произвольного доступа Паскаля позволяют вызывать компоненты в любом порядке по их номеру.

Важной особенностью файлов является то, что данные, содержащиеся в файле, переносятся на внешние носители. Файловый тип Паскаля – это единственный тип значений, посредством которого данные, обрабатываемые программой, могут быть получены извне, а результаты могут быть переданы во внешний мир. Это единственный тип значений, который связывает программу с внешними устройствами ЭВМ.

Работа с файлами в Паскале

Любой файл имеет три характерные особенности. Во-первых, у него есть имя, что дает возможность программе работать одновременно с несколькими файлами. Во-вторых, он содержит компоненты одного типа. Типом компонентов может быть любой тип Паскаля, кроме файлов. Иными словами, нельзя создать «файл файлов». В-третьих, длина вновь создаваемого файла никак не оговаривается при его объявлении и ограничивается только емкостью устройств внешней памяти.

Файловый тип или переменную файлового типа в Паскале можно задать одним из трех способов:

Type <имя_ф_типа>=file of<тип_элементов>;
<имя_ф_типа>=text;
<имя_ф_типа>=file;

Здесь <имя_ф_типа> – имя файлового типа (правильный идентификатор); File, of – зарезервированные слова (файл, из); <тип_элементов> – любой тип Паскаля, кроме файлов.

Пример описания файлового типа в Паскале

```
Type
   Product= record
Name: string;
      Code: word;
   End;
   Text80= file of string[80];
Var
   F1: file of char;
```

F2: text;
F3: file;
F4: Text80; F5: file of Product;

В зависимости от способа объявления можно выделить три вида файлов Паскаля:

- типизированные файлы Паскаля(задаются предложением file of..);
- текстовые файлы Паскаля(определяются типом text);
- нетипизированные файлы Паскаля(определяются типом file).

Следует помнить, что физические файлы на магнитных дисках и переменные файлового типа в программе на Паскале – объекты различные. Переменные файлового типа в Паскале могут соответствовать не только физическим файлам, но и логическим устройствам, связанным с вводом/выводом информации. Например, клавиатуре и экрану соответствуют файлы со стандартными именами Input, Output.

Как известно, каждый тип данных в Паскале, вообще говоря, определяет множество значений и множество операций над значениями этого типа. Однако над значениями файлового типа Паскаля не определены какие-либо операции, в том числе операции отношения и присваивания, так что даже такое простое действие, как присваивание значения одной файловой переменной другой файловой переменной, имеющей тот же самый тип, запрещено. Все операции могут производиться лишь с элементами (компонентами) файлов. Естественно, что множество операций над компонентами файла определяется типом компонент.

Переменные файлового типа используются в программе только в качестве параметров собственных и стандартных процедур и функций.

Основные процедуры и функции для работы с файлами

1. До начала работы с файлами в Паскале необходимо установить связь между файловой переменной и именем физического дискового файла:

Assign(<файловая_переменная>, <имя_дискового_файла>)

Следует помнить, что имя дискового файла при необходимости должно содержать путь доступа к этому файлу, включая имя дисковода. При этом имя дискового файла – строковая величина, т.е. должна быть заключена в апострофы. Например:

Пример процедуры Assign в Паскале

Assign (chf, 'G:\Home\ Student\ Lang\ Pascal\ primer.dat');

Если путь не указан, то программа будет искать файл в своем рабочем каталоге и по указанным путям в autoexec.bat.

Вместо имени дискового файла можно указать имя логического устройства, каждое из которых имеет стандартное имя:

CON – консоль, т.е. клавиатура-дисплей;

PRN – принтер. Если к компьютеру подключено несколько принтеров, доступ к ним осуществляется по именам LPT1, LPT2, LPT3.

Не разрешается связывать с одним физическим файлом более одной файловой переменной.

2.После окончания работы с файлами на Паскале, они должны быть закрыты.

Close(<список файловых переменных>);

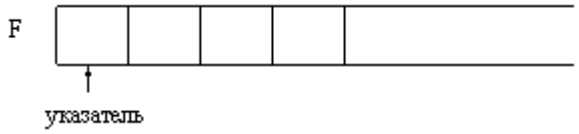
При выполнении этой процедуры закрываются соответствующие физические файлы и фиксируются сделанные изменения. Следует иметь в виду, что при выполнении процедуры close связь файловой переменной с именем дискового файла, установленная ранее процедурой assign, сохраняется, следовательно, файл можно повторно открыть без дополнительного использования процедуры assign.

Работа с файлами заключается, в основном, в записи элементов в файл и считывании их из файла. Для удобства описания этих процедур введем понятие указателя, который определяет позицию доступа, т.е. ту позицию файла, которая доступна для чтения (в режиме чтения), либо для записи (в режиме записи). Позиция файла, следующая за последней компонентой файла (или первая позиция пустого файла) помечается специальным маркером, который отличается от любых компонент файла. Благодаря этому маркеру определяется конец файла.

3.Подготовка к записи в файл Паскаля

Rewrite(<имя_ф_переменной>);

Процедура Rewrite(f) (где f – имя файловой переменной) устанавливает файл с именем f в начальное состояние режима записи, в результате чего указатель устанавливается на первую позицию файла. Если ранее в этот файл были записаны какие-либо элементы, то они становятся недоступными. Результат выполнения процедуры rewrite(f); выглядит следующим образом:



4.Запись в файл Паскаля

Write(<имя_ф_переменной>, <список записи>);

При выполнении процедуры write(f, x) в ту позицию, на которую показывает указатель, записывается очередная компонента, после чего указатель смещается на следующую позицию. Естественно, тип выражения x должен совпадать с типом компонент файла. Результат действия процедуры write(f, x) можно изобразить так:

Состояние файла f до выполнения процедуры

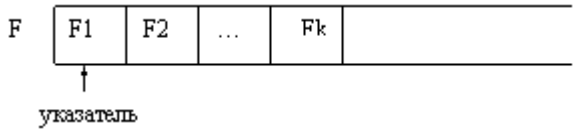
Состояние файла f после выполнения процедуры

Для типизированных файлов выполняется следующее утверждение: если в списке записи перечислено несколько выражений, то они записываются в файл, начиная с первой доступной позиции, а указатель смещается на число позиций, равное числу записываемых выражений.

5.Подготовка файла к чтению Паскаля

Reset(<имя_ф_переменной>);

Эта процедура ищет на диске уже существующий файл и переводит его в режим чтения, устанавливая указатель на первую позицию файла. Результат выполнения этой процедуры можно изобразить следующим образом:



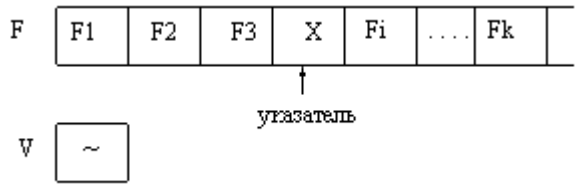
Если происходит попытка открыть для чтения не существующий еще на диске файл, то возникает ошибка ввода/вывода, и выполнение программы будет прервано.

6.Чтение из файла в Паскале

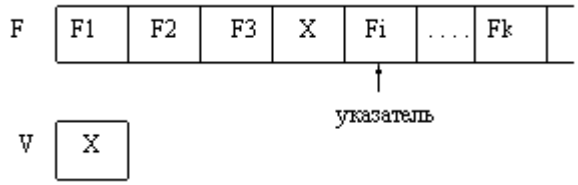
Read(<имя_ф_переменной>,<список переменных>);

Рассмотрим результат действия процедуры read(f, v):

Состояние файла f и переменной v до выполнения процедуры:



Состояние файла f и переменной v после выполнения процедуры:



Для типизированных файлов при выполнении процедуры read() последовательно считывается, начиная с текущей позиции указателя, число компонент файла, соответствующее числу переменных в списке, а указатель смещается на это число позиций.

В большинстве задач, в которых используются файлы, необходимо последовательно перебрать компоненты и произвести их обработку. В таком случае необходимо иметь возможность определять, указывает ли указатель на какую-то компоненту файла, или он уже вышел за пределы файла и указывает на маркер конца файла.

7.Функция определения достижения конца файла в Паскале

Eof(<имя_ф_переменной>);

Название этой функции является сложносокращенным словом от end of file. Значение этой функции имеет значение true, если конец файла уже

достигнут, т.е. указатель стоит на позиции, следующей за последней компонентой файла. В противном случае значение функции – false.

8.Изменение имени файла в Паскале

Rename(<имя_ф_переменной>, <новое_имя_файла>);

Здесь новое_ имя_ файла – строковое выражение, содержащее новое имя файла, возможно с указанием пути доступа к нему.

Перед выполнением этой процедуры необходимо закрыть файл, если он ранее был открыт.

9.Уничтожение файла в Паскале

Erase(<имя_ф_переменной>);

Перед выполнением этой процедуры необходимо закрыть файл, если он ранее был открыт.

10.Уничтожение части файла от текущей позиции указателя до конца в Паскале

Truncate(<имя_ф_переменной>);

11.Файл Паскаля может быть открыт для добавления записей в конец файла

Append(<имя_ф_переменной>);

Типизированные файлы Паскаля. Длина любого компонента типизированного файла строго постоянна, т.к. тип компонент определяется при описании, а, следовательно, определяется объем памяти, отводимый под каждую компоненту. Это дает возможность организовать прямой доступ к каждой компоненте (т.е. доступ по порядковому номеру).

Перед первым обращением к процедурам ввода/вывода указатель файла стоит в его начале и указывает на его первый компонент с номером 0. После каждого чтения или записи указатель сдвигается к следующему компоненту файла. Переменные и выражения в списках ввода и вывода в процедурах **read() и write()** должны иметь тот же тип, что и компоненты файла Паскаля. Если этих переменных или выражений в списке несколько, то указатель будет смещаться после каждой операции обмена данными на соответствующее число позиций.

Для облегчения перемещения указателя по файлу и доступа к компонентам типизированного файла существуют специальные процедуры и функции:

fileSize(<имя_ф_переменной>) – функция Паскаля, определяющая число компонентов в файле;

filePos(<имя_ф_переменной>) – функция Паскаля, значением которой является текущая позиция указателя;

seek(<имя_ф_переменной>,n) – процедура Паскаля, смещающая указатель на компоненту файла с номером n. Так, процедура seek(<имя_ф_переменной>,0) установит указатель в начало файла, а процедура seek(<имя_ф_переменной>, FileSize(<имя_ф_переменной>)) установит указатель на признак конца файла.

Текстовые файлы Паскаля. Текстовые файлы предназначены для хранения текстовой информации. Именно в таких файлах хранятся, например,

исходные тексты программ. Компоненты текстовых файлов могут иметь переменную длину, что существенно влияет на характер работы с ними. Доступ к каждой строке текстового файла Паскаля возможен лишь последовательно, начиная с первой. К текстовым файлам применимы процедуры assign, reset, rewrite, read, write и функция eof. Процедуры и функции seek, filepos, filesize к ним не применяются. При создании текстового файла в конце каждой записи (строки) ставится специальный признак EOLN(end of line – конец строки). Для определения достижения конца строки существует одноименная логическая функция EOLN(<имя_ф_переменной>), которая принимает значение true, если конец строки достигнут.

Форма обращения к процедурам write и read для текстовых и типизированных файлов одинакова, но их использование принципиально различается.

В списке записываемых в текстовый файл элементов могут чередоваться в произвольном порядке числовые, символьные, строковые выражения. При этом строковые и символьные элементы записываются непосредственно, а числовые из машинной формы автоматически преобразуются в строку символов.

- текстовые файлы удобнее для восприятия человеком, а типизированные соответствуют машинному представлению объектов;
- текстовые файлы, как правило, длиннее типизированных;
- длина текстовых файлов зависит не только от количества записей, но и от величины переменных.

Так, в типизированном файле числа 6, 65 и 165 как целые будут представлены одним и тем же числом байт. А в текстовых файлах, после преобразования в строку, они будут иметь разную длину. Это вызывает проблемы при расшифровке текстовых файлов. Пусть в текстовый файл пишутся подряд целые числа (типа byte): 2, 12, 2, 128. Тогда в файле образуется запись 2122128. При попытке прочитать из такого файла переменную типа byte программа прочитает всю строку и выдаст сообщение об ошибке, связанной с переполнением диапазона.

Но, вообще-то, такой файл не понимает не только машина, а и человек.

Чтобы избежать этой ошибки, достаточно вставить при записи в файл после каждой переменной пробел. Тогда программа при каждом чтении берет символы от пробела до пробела и правильно преобразует текстовое представление в число.

Кроме процедур read и write при работе с текстовыми файлами используются их разновидности readln и writeln. Отличие заключается в том, что процедура writeln после записи заданного списка записывает в файл специальный маркер конца строки. Этот признак воспринимается как переход к новой строке. Процедура readln после считывания заданного списка ищет в файле следующий признак конца строки и подготавливается к чтению с начала следующей строки.

Пример решения задачи с файлами Паскаля

Пусть нам необходимо сформировать текстовый файл с помощью Паскаля, а затем переписать из данного файла во второй только те строки, которые начинаются с буквы «А» или «а».

Пояснения: нам понадобятся две файловые переменные f1 и f2, поскольку оба файла текстовые, то тип переменных будет text. Задача разбивается на два этапа: первый – формирование первого файла; второй – чтение первого файла и формирование второго.

Для завершенности решения задачи есть смысл добавить еще одну часть, которая в задаче явно не указана – вывод на экран содержимого второго файла.

Пример процедуры Assign Паскаля

```
Program primer;
Var f1,f2:text;
   I,n: integer;
   S: string;
Begin
{формируем первый файл}
   Assign(f1, ‘file1.txt’); {устанавливаем связь файловой переменной с
физическим файлом на диске}
   Rewrite(f1);  {открываем файл для записи}
   Readln(n) {определим количество вводимых строк}
   for i:=1 to n do
   begin
      readln(s); {вводим с клавиатуры строки}
      writeln(f1,s); {записываем последовательно строки в файл}
   end;

  close(f1); {заканчиваем работу с первым файлом, теперь на диске существует
файл с именем file1.txt, содержащий введенные нами строки. На этом
программу можно закончить, работу с файлом можно продолжить в другой
программе, в другое время, но мы продолжим}
{часть вторая: чтение из первого файла и формирование второго}
   Reset(f1); {открываем первый файл для чтения}
     Assign(f2, ‘file2.txt’); {устанавливаем связь второй файловой
переменной с физическим файлом}
   Rewrite(f2); {открываем второй файл для записи}
{Дальше необходимо последовательно считывать строки из первого
файла, проверять выполнение условия и записывать нужные строки во второй
файл. Для чтения из текстового файла рекомендуется использовать цикл по
условию «пока не конец файла»}
   While not eof(f1) do
   Begin
      Readln(f1,s);{считываем очередную строку из первого файла}
       If (s[1]=’A’) or (s[1]=’a’) then  Writeln(f2,s); {записываем во второй
файл строки, удовлетворяющие условию}
   End;
   Close(f1,f2); {заканчиваем работу с файлами}
 {часть третья: выводим на экран второй файл}
   Writeln;
   Writeln(‘Второй файл содержит строки:’);
```

```
    Reset(f2); {открываем второй файл для чтения}
    While not eof(f2) do {пока не конец второго файла}
    Begin
      Readln(f2,s);{считываем очередную строку из второго файла}
      Writeln(s); {выводим строку на экран}
    End;
End.
```

Контрольные вопросы:

1. Что такое файл?
2. Какие типы файлов вы знаете?
3. Дайте характеристику файлов доступа (последовательному и произвольному).
4. Перечислите операторов для работы с текстовыми файлами и их назначение.
5. Сравните действия операторов с последовательным и произвольным доступом.