ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ
кафедра ИУС ФТК СПбГПУ

- Веренинов Игорь Андреевич

- Лекция 8

Особенности ввода-вывода массивов .

- два принципиально различающихся случая ввода двумерных символьных массивов в оперативную память:
- 1.все строки выровнены по правой границе ;
- 2.строки имеют разную длину во входном файле.

Ввод строк постоянной длины

```
Program formatmas;
TYPE
T=ARRAY [1..5,1..10] OF CHAR;
VAR
A:T; I,J:BYTE;
F:TEXT;   {файловая переменная текстового типа}
  Begin
     ASSIGN (F,'dan.inp'); {связь переменной F с файлом}
     RESET (F);  {подготовка к чтению}
For I:=1 to 5 do  {цикл перебора строк }
        Begin
   For j:=1 to 10 do  {цикл ввода  строки символов}
  Read (F,A[I,J]);
   Readln (F)   {перевод строки входного файла}
         End;
Close (F);  {закрытие входного файла}
       For I:=1 to 5 do   {перебор строк при выводе на экран}
                Begin
       For j:=1 to 10 do  {цикл вывода одной строки}
         Write (A[I,J]);
         Writeln    {переход на другую строку}
                  End;
   End.   {конец текста}
```

Ввод массива строк переменной длины

```
Program  nonformat;
TYPE
T=ARRAY [1..5,1..10] OF CHAR;
VAR
A:T; I,J:BYTE;
F:TEXT;   {файловая переменная текстового типа}
  Begin
     ASSIGN (F,'dan.inp'); {связь переменной F с файлом}
     RESET (F);  {подготовка к чтению}
For I:=1 to 5 do  {цикл перебора строк }
        Begin
J:=1;
   WHILE  not EOLN (F) do  {цикл ввода  строки символов}
        Begin
     Read (F,A[I,J]);
      J:=J+1
         End;
   Readln (F)   {перевод строки входного файла}
         End;
Close (F);  {закрытие входного файла}
       For I:=1 to 5 do   {перебор строк при выводе на экран}
                Begin
       For j:=1 to 10 do  {цикл вывода одной строки}
         Write (A[I,J]);
         Writeln    {переход на другую строку}
                  End;
   End.   {конец текста}
```

Процедуры и функции .

- **Глобальные, локальные, формальные и фактические переменные (1 пример).**

```
Program a1;
VAR
X:REAL;  {глобальная переменная}
  Procedure CHANGE;
Begin
X:=1.0   {используемая внутри процедуры глобальнаяпеременная}
End; {конецпроцедуры}
Begin   {начало основной программы}
X:=0.0;  {на место глобальной переменной записали нуль}
  CHANGE;  {обращение к процедуре}
 WRITELN('X=',X);  {значение X будет равно 1.0}
End.
```

Процедуры и функции

```
Program a2; {2 пример}
VAR
X:REAL;  {глобальная переменная}
  Procedure CHANGE;
VAR
X:REAL;  {локальная переменная}
Begin
```

- Глобальные, локальные, формальные и фактические переменные (2 пример).

```
X:=1.0   {используемая внутри процедуры локальная переменная}
End; {конец процедуры}
Begin   {начало основной программы}
X:=0.0;  {на место глобальной переменной записали нуль}
  CHANGE;  {обращение к процедуре}
 WRITELN('X=',X);  {значение глобального X будет равно 0.0}
End.
```

Процедуры и функции

- **Глобальные, локальные, формальные и фактические переменные (3 пример-передача по значению).**

```
Program a3;
VAR
X:REAL;  {глобальная переменная}
  Procedure CHANGE(Y:REAL);
Begin
Y:=1.0   {используемая внутри процедуры формальнаяпеременная}
End; {конец процедуры
Begin   {начало основной программы}
X:=0.0;  {на место глобальной переменной записали нуль}
  CHANGE (X);  {обращение к процедуре}
 WRITELN('X=',X);  {значение X будет равно по-прежнему 0.0}
End
```

Процедуры и функции

- **Глобальные, локальные, формальные и фактические переменные (4 пример-передача по имени).**

```
Program a4;
VAR
X:REAL;  {глобальная переменная}
  Procedure CHANGE(VAR Y:REAL);
Begin
Y:=1.0   {используемая внутри процедуры формальнаяпеременная}
End; {конец процедуры
Begin   {начало основной программы}
X:=0.0;  {на место глобальной переменной записали нуль}
  CHANGE (X);  {обращение к процедуре}
 WRITELN('X=', X);  {значение X будет равно 1.0}
End.
```

Процедуры и функции

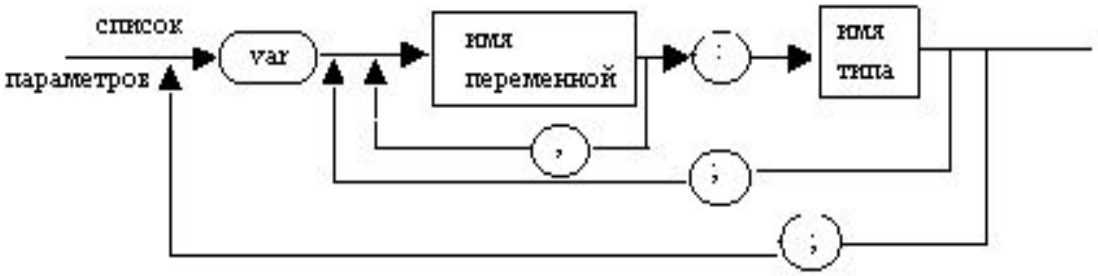
Области действия имен.

```
Program a1 (……..);
VAR
A,B:REAL;
Procedure a2 (……..);
   VAR
   C,D:INTEGER;
      Procedure a3 (…….);
        VAR
         E,F: CHAR;
          Begin  {исполняемая часть a3}
             .{можно использовать A,B,C,D,E,F}
             .
             .
           End;  {конец процедуры a3}
   Begin   {исполняемая часть a2}
         .{можно использовать A,B,C,D}
         .{можно вызывать процедуру a3}
         .
     End;   {конец  a2}
  Begin  {начало головной программы}
    . {здесь можно использовать только  A и B}
      {и вызывать процедуру a2}
     .  {нельзя вызывать a3}
     .
   End.  {конец головной программы}
```

Процедуры

- PROCEDURE <имя> (<список формальных параметров>); {это заголовок}
- {область описания такой же структуры , как и в основной программе}
- Begin
- {область исполняемых операторов}
- End.

Процедуры

- Синтаксическая диаграмма списка формальных параметров имеет вид:
- 

Особенности списка формальных параметров:

- 1.Все выходные параметры должны быть с описателем VAR , т.е. передаваться по имени;
- 2.Если параметр является одновременно и входным, и выходным, то VAR также необходим;
- 3.Имена типов параметров должны быть либо базовыми, либо описаны выше. Нельзя использовать явные описания производных типов, например, ARRAY [1..3,1..7] OF REAL, а можно только имена типов.

Особенности списка формальных параметров:

- 4.Громоздкие входные параметры , например, большие массивы передавать по имени. При этом в машинный стек при вызове процедуры записывается короткий адрес фактического параметра , а при передаче по значению (без описателя VAR) необходимо записывать все значения этого громоздкого данного.
- 5.Резервирование оперативной памяти для фактических параметров осуществляется при описании типа этого параметра , исходя из его возможных максимальных размеров . Исключение составляют так называемые открытые массивы .

Пример процедуры

```
Program ver1;
    TYPE
     M=ARRAY [1..10] OF INTEGER;
   VAR
A:M;  I ,S: INTEGER;
PROCEDURE SUM (VAR A1:M ; N: INTEGER; VAR S1: INTEGER);
VAR
    I: INTEGER;
 Begin
     S1:=0;
       For I:=1 to N do
S1:=S1+A[I];
WRITELN ('S1=',S1,'********');
End;  {конец процедуры}
  Begin    {начало головной программы}
ASSIGN (input, ' dan.inp');
RESET  (input);
     For I:=1 to 10 do
Begin
   Read (A[I]);  {чтение массива из 10 чисел из файла}
   Writeln (A[I]);  {вывод на экран по одному числу на строке}
End;
   SUM (A,5,S);  {обращение к процедуре}
    Writeln( 'S=',S);
  Close(input)
End.   {конец текста}
```

Функции.

- **FUNCTION <имя функции> (<список формальных параметров>) :<имя типа значения функции>;**
- **Особенности:**
- **1.Имя функции должно приобретать какое-либо значение в исполняемой части блока функции;**
- **2.Основной результат обычно формируется на месте имени функции;**
- **3.Обращение к функции осуществляется упоминанием ее имени в выражении (это основной способ использования функций) , а также непосредственно вызывая ее , как процедуру. В последнем случае значение, сформированное на месте имени функции не используется , а применение функции в таком контексте имеет смысл лишь в том случае, если используется ее побочный результат, возвращаемый в виде значения какого-либо выходного параметра.**

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ
кафедра ИУС ФТК СПбГПУ

- Веренинов Игорь Андреевич
- Лекция 9

Использование открытых массивов.

Открытый массив- это одномерный массив , размер которого в процедуре заранее не определен .

```
Program testopenmas;
TYPE
t=array [1..3] of  integer;
 VAR
as: t;
i, j :integer;
Procedure p (var a1: array of integer);
  VAR
i , j: integer;
Begin
{цикл чтения при изменении индекса массива a1 от нижнего зарезервированного значения  до верхнего}
for i:=low(a1) to high(a1) do
Begin
readln (a1[i])
end;
End;{конец процедуры}
```

Использование открытых массивов

```
Begin  {начало головной программы}
assign(input,'dan12.txt');
 reset(input);
 p(as);   {обращение к процедуре}
close(input);
{вывод  элементов массива as без последнего}
For i:=low(a) to high(a)-1 do
Writeln (as[i] :2);
end.
```

- особенности этого текста:
- 1.В тексте процедуры в списке формальных параметров отсутствует указание на размер массива и нет имени типа t , т.е. массив считается открытым.
- 2.При организации цикла чтения данных в массив используются функции **low** и **high** , которые при обращении к этой процедуре возвратят начальное значение индекса фактического массива as и конечное его значение (1 и 3 соответственно).
- 3.В головной программе будет выводиться на экран только два значения массива as[1] и as[2].

Значение индексов открытых формальных массивов начинается от 0

Использование открытых массивов

Изменим текст:

```
 Program testopenmas1;
TYPE
t=array [1..3] of  integer;
 VAR
as: t;
i, j :integer;
Procedure p (var a1: array of integer);
  VAR
i , j: integer;
Begin
{цикл чтения при изменении индекса массива a1 от нижнего зарезервированногозначения  до
      верхнего}
for i:=1 to 3 do
Begin
readln (a1[i])
end;
End;{конец процедуры}
```

Использование открытых массивов

- Begin {начало головной программы}
- assign(input,'dan12.txt');
- reset(input);
- p(as); {обращение к процедуре}
- close(input);
- {вывод элементов массива as без первого}
- For i:=1 to 3 do
- Writeln (as[i] :2);
- end.
- Если во входном файле записаны числа 1,2,3,4,5 по одному на строке, то в процедуре первые 3 числа попадут на место a1[1], a1[2],a1[3] соответственно . А при передаче по имени массива as адрес as[1] передается вместо адреса a1[0], в который ничего не введется . Поэтому выведется значение as[1]=0, as[1]=1, as[2]=2.

Процедурный тип данных.

- 1.Разработать процедуру вычисления таблицы значений произвольной вещественной функции одной вещественной переменной в зависимости от значений ее аргумента X, изменяющегося в заданном диапазоне AF<=X<=BF с заданным шагом изменения H .
- 2. Применить эту функцию для составления таблицы значений функций F1(X)=X*X*X+1 для 5.0<=X<=10.0 с шагом H=1 и F2(X)=cos(X) для 1.0<=X<=2.0 с шагом H=0.1.

Процедурный тип данных.

```
Program PPP;
TYPE
tf=function (X:real):real;   {имя процедурного типа}
VAR ff:real;
{$F+}    {опция компилятора}
function F1(x:real):real;
Begin F1:=X*X*X+1
 end; { конец функции }
function F2(x:real):real;
Begin F2:=cos(x)
 End; { конец функции}
{$F-}    {отмена предыдущей опции компилятора}
Procedure TABLICA(AF, BF,H:real;IF1:tf;var FUN : real);
     VAR
I,N:integer; X:real;
Begin
N:=trunc((BF-AF+H/2)/H)+1;{вычисление длины таблицы}
X:=AF;
For I:=1 to N do
Begin
fun:=IF1(x); writeln (x:4:1,fun:8:2);
X:=X+H
End;
End;{конец процедуры}
```

Процедурный тип данных.

```
BEGIN {головная программа }
tablica(5.0,10.0,1.0,f1,ff);
tablica(1.0,2.0,0.1,f2,ff)
            END.
```

Особенности этого текста в следующем:

1. Одним из формальных параметров процедуры TABLICA является имя функции IF1.
2. Опция компилятора {$F+} должна применяться всегда в случае использования процедур (функций) в качестве фактических параметров перед описанием их текста.

В результате выполнения этой программы на экране получим результат:

```
 126.00 {значение функции x*x*x+1 при x=5}
……
10.0 1001.00
0.54  {косинус одного радиана}
 ……
-0.42
```

Символьные строки (переменные типа STRING).

VAR
A, B : STRING[5];
C : STRING;
С данными типа **STRING** можно проводить следующие операции:
1.Операции присваивания :
A:='IVANOV';
Writeln(A); {будет выведена строка IVANO , а буква V будет усечена , так как превышает допустимую длину символьной строки A.
2.Операции сложения (конкатенации) строк ;
C:='IVANOV';
B:= 'PETR ';
C:= B+C;
Writeln(C); {выведется строка PETR IVANOV, при этом в нулевой байт запишется число =11, равное длине новой строки C}
3. Операции сравнения двух строк, например, A>=B ,C<A и т.д.
4.Операции ввода-вывода.
5. переменные типа STRING можно индексировать , например, можно писать
A[3] :=C[2].

Символьные строки (переменные типа STRING).

6.если посимвольно читать строку из текстового файла , а затем работать со строкой как с единым целым, то возникнет ошибка , так как при этом не будет автоматически формироваться байт длины. Ниже приведен такой пример с учетом описаний A,B,C .

```
For I:=1 to 5 do
Read(A[i]);
Readln(B);
IF A>B THEN ……..
```

Для работысо строками наиболее часто используются функции COPY (A, NB,L), VAL (A,X,CODE) и процедура INSERT (SUBST,A,NB).

Записи (RECORD).

- Пусть во входном файле находится 20 строк вида:
- IVANOV P.I. 1987 P
- Число символов (10 4 2 4 11)
- Program ZAPIS;
- TYPE
- ZAP=RECORD
- FAM:string[10];
- INIT: string [4];
- GOD: integer;
- REG:char
- END;
- VAR
- B:array [1..20] of zap;
- I:integer;
- C:char;
-

Записи (RECORD).

```
Begin {ввод и вывод массива записей}
    ASSIGN (input,'dan.inp');
     RESET(input);
      ASSIGN(output,'res.out');
       Rewrite(output);
  For I:=1 to20 do
     Begin
    Readln (b[I].fam , b[I].init ,b[I].god, c, b[I].reg );
  WITH b[I] do
   Writeln (fam,init:5, god:6,reg:2)
      End;
Close (input);
 Close (output)
       End.
```

Записи (RECORD).

- 1.Для работы с полем записи указывается ее имя , а затем через точку имя поля записи.
- 2.По мере чтения строки символов из файла указатель файла перемещается вправо на одну позицию (символ), и программист должен следить за правильностью чтения полей записи в соответствии с их типом.
- 3. Программа работает с массивом записей B, который индексируется с помощью переменной I.
- 4. Чтобы избегать постоянного повторения общего имени элемента массива записей B[I] , в языке PASCAL предусмотрен оператор WITH
- 5. Вывод в текстовый файл и ввод из него нельзя проводить сразу со всей записью , т.е. нельзя писать WRITELN(B[I])

Записи (RECORD).

- С записями можно производить следующие операции:
- 1.Операция присвоения может выполняться применительно ко всей записи целиком, например, можно записать B[2]:=B[3], и при этом все поля третьего элемента массива B запишутся на место полей второго элемента.
- 2.С полями записей разрешается проводить все операции в соответствии с их типом.