**Лекция №1**

**Использование модуля CRT.**

Модуль CRT увеличивает возможности текстового ввода / вывода. Дополнительные возможности управления клавиатурой реализуются двумя функциями: KeyPressed и ReadKey.

Функция KeyPressed возвращает значение типа Boolean, указывающее состояние буфера клавиатуры: а) False, если буфер пуст (никакая клавиша не была нажата); б) True, если в буфере есть хотя бы один символ (нажата любая клавиша клавиатуры).

Функция ReadKey возвращает значение типа CHAR при анализе буфера клавиатуры. Значением функции является символ нажатой клавиши. Если клавиша не нажимается, то функция работает в режиме ожидания нажатия клавиши.

Пример:

Uses CRT;

Var c: char;

Begin

Repeat

C:= ReadKey;

If C <> #0 then writeln (ord (C) )

Else writeln (‘0’, ord (ReadKey): 8)

Until C = #27; {#27 – “ESC”}

End.

Печать расширенного кода любой клавиши, нажатие “ESC” – выход из программы.

На практике в любой программе в конце перед “END.” ставят оператор READLN без параметров, требующего от пользователя нажатие клавиши “Enter”. Пока пользователь не нажал “Enter”, он может смотреть результат работы программы (без “Alt+F5”).

Процедура TextColor (Color: Byte) определяет цвет символов, а процедура TextBackGround (Color: Byte) – цвет фона экрана (0 – это чёрный, а 15 – белый).

Процедура ClrScr очищает экран или окно (Window), т.е. экран заполняется цветом фона, а курсор устанавливается в его левый верхний угол, например:

Uses CRT;

Var C: char;

Begin

TextBackGround (red);

ClrScr; {заполняем экран красным цветом}

Writeln (‘Нажмите любую клавишу …’);

C:= ReadKey; {Ждём нажатия любой клавиши}

TextBackGround (Black);

ClrScr; {Восстанавливаем чёрный фон экрана}

End.

Процедура Window определяет текстовое окно (часть экрана), курсор помещается в левый верхний угол окна, а само окно заполняется цветом фона (очищается), например:

Window (x1, y1, x2, y2);

Здесь x1, y1 – координаты левого верхнего угла, x2, y2 - координаты правого нижнего угла. Левый верхний угол (1, 1), горизонтальная координата увеличивается слева направо, а вертикальная – сверху вниз.

Процедура GOTOXY (x, y: Byte) переводит курсор в нужное место экрана или текущего окна, например:

GOTOXY (100, 50);

GOTOXY (i+5, 2\*j+10);

Процедура Sound (F: Word) включает звук с частотой F в герцах. Звук будет работать до вызова процедуры NoSound. NoSound выключает звук.

Процедура Delay (T: Word) задерживает выполнение следующего оператора программы в течение времени T (в миллисекундах).

Для генерации звукового сигнала обычно используется схема “Sound – Delay – NoSound”, например:

Sound (200);

Delay (1000);

NoSOund;

Задача: вычислить y= a+ b

Program PR1;

Uses CRT;

VAR a, b, y: real; c: char;

Begin

ClrScr;

Write (‘Введи a и b’);

Readln (a, b);

Y:=a + b;

Writeln (‘y=’, y:6:3);

C:=readkey;

End.

Та же задача, программа моргает разным цветом.

Program PR2;

Uses CRT;

VAR a, b, y, i: integer; c: char;

Begin

Window (2, 12, 169, 169);

For i:= 1 to 3 do begin

Textbackground (0);

Clrscr;

Window (random (80), random (25); random (80), random (25));

Textbackground ( random (15));

Clrscr;

C:= readkey;

Delay (2000);

End;

Textbackground (4);

Clrscr;

Textcolor (2);

Write (‘vvedi a I b ‘);

Readln (a, b);

Y:= a+ b;

Textbackground (2);

Textcolor (21);

Writeln (#7, ‘y= ‘, y:6:3);

C:= readkey;

For i:= 1 to 24 do begin

Sound (349+ 10\* i);

Delay (25000);

NoSOund;

End;

C:= readkey;

End.

**Лекция № 14**

**Использование модуля GRAPH.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ядро | Транзит ОП | Среда BP | Область пользователя | Динамическая память - куча | Видеопамять |

ОП

Процессор Сопроцессор Адаптер Монитор

CGA – Color Graphics Adapter

EGA.BGI – Enhanced (усиленный), Borland Graphic Interface

SVGA (supervideo)

EGAVGA.BGI находится в папке \BP\BGI.

Процедура InitGraph (var Driver, Mode: Integer; Path: String);

Инициирует графический режим работы адаптера (выключает текстовый режим).

Driver – тип графического драйвера,

Mode – режим работы графического адаптера,

Path – путь доступа к файлу драйвера.

Пример:

Program P;

Uses Graph;

Var driver, mode: integer;

Begin

Driver:= detect;

InitGraph (driver, mode, ‘D:\BP\BGI’);

If Graphresult <> 0 then HALT (1); {выход из программы}

Рисование точки заданного цвета N:

Putpixel (x, y, N); Координата (0, 0) – верхний левый угол;

Max X и Y – функции GetMaxx и GetMaxy.

Program P;

…

For i:= 0 to GetMaxx div 2 do begin

Putpixel (I, GetMaxy div 2, red);

Putpixel (I + GetMaxx div 2, GetMaxy div 2, green);

…

Рисование линий – Line (x1, y1, x2, y2);

Цвет линии и её стиль должны быть установлены выше.

LineTo (x2, y2); - рисует линию с текущего положения курсора.

GetX, GetY – определяют текущее положение графического курсора.

Перемещение курсора – MoveTo (x, y: integer); MoveRel (dx, dy: integer);

LineRel (dx, dy); - рисование линии.

Рисование прямоугольника:

Line (10, 10, 200, 10);

LineTo (200, 100);

LineTo (10, 100); {LineRel (-190, 0);}

LineTo (10, 10);

Или Rectangle (10, 10, 200, 100);

Рисование окружности – Circle (x0, y0, R);

Рисование мишени:

For i:= 1 to 10 do

Circle (GetMaxx div 2, GetMaxy div 2, 10\*i);

Рисование дуг: Arc (x0, y0, 90, 270, R); Arc (x0, y0, A1, A2, R);

Рисование эллипса – Ellipse (x0, y0, A1, A2, Rx, Ry);

Заливка фигуры – Floodfill (x, y, N);

Цвет и стиль заливки устанавливаются ранее, x, y – координаты точки внутри фигуры, N – цвет границы области, которую красим. Если точка находится в замкнутом контуре фигуры, то заливается фигура. Если - вне фигуры, то закрашивается весь экран, кроме замкнутой фигуры.

Пример:

Program P;

Uses CRT, Graph;

Var R, D: integer;

Begin

D:= detect;

InitGraph (D, R, ‘D:\BP\BGI’);

Setlinestile (2, 1, 3); {2 – штрих-пунктирная, 1 – стандартная, 3 – толщина}

Setbkcolor (15); {цвет фона}

Setcolor (5); {цвет линий и символов}

Outtext (‘privet’); {вывод текста в графическом режиме}

Settextstyle (2, 0, 32); {2 – номер шрифта, 0 – горизонтальное написание букв, 32 – высота букв}

Outtextxy (150, 150, ‘ХА-ХА’); {вывод текста с заданной позиции курсора}

Setfillstyle (4, 2); {4 – образец штриховки, 2 – номер цвета}

Circle (250, 250, 90);

Floodfile (100, 100, 4);

Delay (15000);

**Лекция № 15**

**Модули**

Максимальная длина программного сегмента в Pascal 64 Кбайта. Если требуется разрабатывать большие программы, то можно использовать библиотеки с текстами процедур и модули. Текст файла можно включить в программу директивой компилятора $I.

Модули удобны для построения собственных библиотек процедур и функций, а также создания программ любого размера. Модули предварительно компилируются независимо от использующей их программы. Результатом компиляции модуля является файл с расширением .TPU, а чтобы вызвать его, необходимо указать его имя в Uses.

Unit <имя>;

<Описание>

End.

Program P;

Uses <имя>;

…

End.

Схема сборки:

Unit <имя>; Program P; Compile <имя>.TPU; <P>.OBJ Link

<P>.EXE выполнение

Структура модуля

Unit <имя>;

INTERFACE

<описания> для связи с внешним миром

IMPLEMENTATION

<описания> начинка модуля: описания внутренних переменных и описания всех процедур и функций модуля - исполнительная часть модуля

Begin

<операторы> инициирующая часть модуля (необязательная)

End.

Модуль состоит из заголовка и трёх составных частей, любая из которых может быть пустой.

Пример:

Unit modd;

Interface

Const d=0.8;

Var a: string;

Implementation

…

Begin

A:= ‘HELLO’;

End.

Program p;

Uses modd;

Var b, y: real;

Begin

Writeln (a);

Y:= d+1;

Writeln (y);

End.

В интерфейсной части описываются только заголовки процедур.

Unit matem;

Interface

Procedure Add (a, b: real; var y: real);

Procedure Sub (a, b: real; var y: real);

Function Mult (a, b: real): real;

Function Divide (a, b: real): real;

Implementation

Procedure Add;

Begin

Y:= a+ b;

End;

Procedure Sub;

Begin

Y:= a- b;

End;

Function Mult;

Mult:= a\* b;

End;

Function Divide;

Divide:= a/b;

End;

End.

Пример: W= (2,5 a b + c) / sin (x2/4)

Program P;

Uses crt, matem;

Var a, b, c, x, W: real;

Begin

Clrscr;

Write (‘Введи a, b, c, x’);

Readln (a, b, c, x);

Add ( Mult (Mult (2.5, a), b), c, W);

W:= Divide (W, sin (Divide (Mult (x, x), 4)));

Writeln (W: 10: 6);

End.

**Лекция № 16**

**Сложные типы данных**

Тип данных предполагает способ записи данных, их упорядоченность между собой, операции, осуществляемые над ними.

Типы данных делятся на:

1. Простые;
2. Структурные;
3. Указатель (ссылка);
4. Процедурный тип.

Простые типы делятся на:

А) вещественные;

Б) порядковые.

Порядковые – это:

1. Целочисленные
2. Перечислимый тип
3. Тип – диапазон
4. Символьный тип
5. Логический тип

Структурные типы данных делятся на:

1. Массивы
2. Строки
3. Записи
4. Файлы
5. Множества
6. Объекты

Порядковый тип:

К любому из этих типов применима функция, определяющая порядковый номер всех значений X:

ORD (X), например: ORD (5) = 5; ORD (TRUE) = 1; ORD (‘A’) = <двоичный код буквы>

Функция, вычисляющая предыдущее значение:

PRED (X), например: ORD (PRED (X)) = ORD (X) -1

И последующее значение SUCC (X), например: ORD (SUCC (X)) = ORD (X) + 1

Тип – диапазон:

1. Var A: array [1 .. 100] of char; [1 .. 100] – диапазон изменения значения
2. Type B = 1 .. 100;

Var C, D: B; E: integer;

{$R+} – опция подключения транслятора проверки размерности массива

Begin

E:= D;

E:= 101;

C:= E; - ошибка

В обычном варианте такая проверка не осуществляется.

Var F: 1 .. 10;

Символьный тип данных – char

ASCII (American Standart Code for information interchange) – американский стандартный код для обмена информацией – это 7-битовый код: 128 символов (латинские буквы, цифры, …) – 0 .. 127

Используется и 8-битовый код (расширенный) – 256 символов (с 0).

‘0’ < ‘1’ < … < ‘9’ < ‘A’ < ‘B’ < …

Русские буквы буквами не называются, символы располагаются не по порядку, также есть пустые символы.

Символьный код:

0 - #48

1 - #49

.

.

.

9 - #57

Операции над символами:

Var A, B, C: char;

IF B <> ‘F’ then …

Эти переменные можно объединять в массивы:

Var M: array [1 .. 100] of char;

Встроенная функция chr (B) преобразует выражение B типа byte в символ и возвращает его своим значением:

#97 = chr (97) = ‘a’

#0 = chr (0) = ‘’ (нулевой символ)

#32 = chr (32) = ‘ ‘ (пробел)

#65 – ‘A’

#27 – ESC

UPCASE (CH) – функция типа char, которая возвращает символ в верхнем регистре, если он определён для аргумента типа char; в противном случае возвращает сам символ CH.

Var c1, c2: char;

Begin

C1:= Upcase (‘s’);

C2:= Upcase (‘ф’);

Writeln (c1, ‘ ‘, c2); - S ф, так как кириллицу не обрабатывает

Шестнадцатиричное число - $ (#36) – диапазон от $00 до $FF

Символьная константа – ‘a’, … , ‘z’ ‘’’’ – символьная константа апостроф

Допускается использовать запись символа путём указания его внутреннего кода, которому предшествует символ # (код 35);

#65 – символ ‘A’

#97 – символ ‘a’

Символы можно записывать в строку (STRING), кроме символа CR (возврат каретки).

Эквивалентны:

#83#121#109#98#111#108 – ‘SYMBOL’

В строку можно вставить CR, если указать #13.

Пример:

#7’ошибка!’#13’нажмите любую клавишу’#7

Пример:

Дан текст, в конце текста стоит #, подсчитать количество букв ‘A”.

1 вариант – простая переменная

Program P;

Var ch: char;

Kol: integer;

Begin

{символы при вводе не разделяются пробелами и не берутся в апострофы}

Kol:= 0;

Read (ch);

While ch <> ‘#’ do

Begin

If (upcase (ch) = ‘A’) then kol:= kol+1;

Read (ch);

End;

Writeln (‘Количество символов A = ‘, kol);

End.

2 вариант (массив)

Program P;

Const n=1000;

Type MAS=array [1 .. n] of char;

Var A: MAS;

I, j, k: integer;

{$B+}

Begin

K:=0;

I:=1;

Read (A[i]);

While (I <= n) and (A[i] <> ‘#’) do

Begin

I:= i+1;

Read (A[i]);

End;

{$B-}

For j:= 1 to I do

If (upcase (A[j]) = ‘A’) then k:= k+1;

Writeln (k);

End.

3 вариант (файл)

Program P;

Var f: text; kol: integer; ch: char;

Begin

Assign (f, ‘name.txt’);

Reset (f);

Kol:= 0:

While not eof (f) do

Begin

Read (f, ch);

If (upcase (ch) = ‘A’) then kol:= kol+1;

{ можно ввести if ch = ‘#’ then break – досрочный выход из файла}

End;

Writeln (kol);

Close (f);

End.

4 вариант (string)

Program P;

Var st: string;

I, k: integer;

Begin

K:= 0;

Readln (st);

For i:= 1 to length (st) do

If (upcase (st[i]) = ‘A’) then k:= k+1;

Writeln (k);

End.

Файлы

Вариант 1:

Program P;

Var f1, f2: text;

N: integer;

Begin

Assign (f1, ‘’);

Assign (f2,’’);

Reset (f1);

Rewrite (f2);

Readln (f1, n);

Writeln(f2, ‘ Privet ‘, ‘n=’, n);

Close (f1);

Close (f2);

End.

Вариант 2:

Program P (input, output);

Var n: integer;

Begin

Assign (f1, ‘’);

Assign (f2,’’);

Reset (input);

Rewrite (output);

Readln(n);

Writeln (‘Privet n= ’, n);

End.

**Лекция №17**

**Строковый тип**

Пример:

Дан текст. В конце текста #. Подсчитать частоту встречаемости буквы А.

А) Program P1;

Const n=1000;

Type mass=array [1 .. n] of char;

Var A: mass; I, j, k: integer;

Begin {Main}

K:= 0; i:= 1;

Read (A[i]);

While (I <= n) and (A[i] <> ‘#’) do

Begin {while}

I:= i+1;

Read (A[i]);

End; {while}

For j:= 1 to I do

If (upcase (A[j]) = ‘A’) then k:= k+1;

Writeln (k);

End. {Main}

Б) Program P2;

Var st: string;

N, I, k: integer; a: string[1];

Begin {main}

Readln (st);

K:= 0;

N:= length (st);

For i:= 1 to n do

Begin

A:= copy (st, I, 1);

If (upcase (a) = ‘A’) then k:= k+1;

End;

Writeln (k);

End. {main}

Если записать

Var c: ‘A .. Z’;

То можно записать и цикл:

For c:= ‘B’ to ‘N’ do S;

Var A: array [‘A’ .. ‘Z’] of integer;

A[‘N’]:= A[‘N’] +1;

Тип строка STRING – строки переменной длины появились в Pascal:

Writeln (‘введи N’); - это строковая константа.

Var ST: string [100];

100 – максимальное число символов, которое понадобится.

В нулевом байте содержится длина, дальше 100 символов (максимальная длина string = 255). Длину st задаёт встроенная функция Length (st) – полностью Function LENGTH (st: string): integer;

Обращение к элементу: ST[i]

Если в описании переменной не указан размер строки, то длина = 255 символов.

Ввод строки в память:

Var s1, st: string [10];

Read (st); {вводить все 10 символов строки без апострофов, а потом Enter}

Readln (st);

Вводить каждую строку отдельно.

ST:= ‘’; {пустая строка – l=0 }

ST:= ST + ‘ABC’; {l=3, + операция CONCAT}

ST:= ST + ‘DEFGHIKL’;

Строки можно сравнивать.

Var s1, s2: string [10];

S1:= ‘PASCAL’;

S2:= ‘TURBO’;

If S1 > S2 then … else {TURBO > PASCAL}

Строки сравнивают по кодам символов.

Функции:

А) Выделение подстроки:

Function COPY (ST: string; POZ: integer; N: integer): string;

Var S, S1: string;

S:= ‘АВТОМОБИЛЬ’;

S1:= COPY (S, 5, 3); {S1=’МОБ’}

Б) CONCAT (ST1: string[; ST2: string; …]): string;

В) LENGTH (ST: string): integer;

Г) Определение позиции первого вхождения подстроки в строку:

POS (Subst: string; ST: string): integer;

K:= POS (‘ТО’, S); {K=3}

Д) Процедура Fillchar (Var S1: string; N: integer; S: string[1]);

Заполнение строки некоторым символом, начиная с некоторой позиции.

1 способ:

Var st: string;

I: integer;

For i:= 1 to 80 do

St[i]:= ‘ ‘; {текущая длина строки = 255 символов}

2 способ:

St := ‘’;

For i:= 1 to 80 do

St:= st + ‘ ‘; { текущая длина = 80}

3 способ:

Fillchar (st[1], 80, ‘ ‘);

St[0]:= chr(80);

Пример: Экзаменационная ведомость

|  |  |  |
| --- | --- | --- |
| № | ФИО | Оценка |
| … | … | … |
| 2. | … | 5 (отл.) |
| 3. | Петров | 2 (неуд.) |

2 сим. 20 сим. 8 сим.

Требуется распечатать список отличников:

Program Sessia;

Var str1: string [30];

FAM: array [1.. 25] of string [20];

I, k, n: integer;

OZ: string [8];

Begin

Readln (N); { количество строк }

K:= 0;

For i:= 1 to n do

Begin

Readln (str1);

OZ:= copy (str1, 23, 8);

If pos (‘5’, OZ) <> 0 then

Begin { отличник }

K:= k + 1;

FAM [k]:= copy (str1, 3, 20);

End;

End;

For i:= 1 to k do

Writeln (FAM [i]);

End.

**Тип данных – запись RECORD**

* Совокупность данных, у которой есть одно имя, расположена в оперативной памяти последовательно, а вместе характеризует объект обработки (студентов, группу).

Студент:

1. ФИО (фамилия, имя, отчество)
2. Год рождения
3. № группы
4. Оценки

Способы объявления:

1 способ:

Type FF= record

FAM: string [15];

IM: string [10];

OT: string [10];

End;

STUD = record

FIO: FF;

GR: 1900 .. 2000;

NGR: string [10];

OZ: array [1.. 4] of 2 .. 5;

End;

Var ST: STUD;

ST1: array [1 ..20] of STUD;

Begin

ST. FIO. FAM := ‘Иванов’;

ST. OZ [1] := 5;

ST1 [10]. OZ [3] := 5;

**Оператор присоединения**

WITH < имя > do S;

< имя > - это имя или список имен;

S – оператор (м.б. составным).

Например: with ST, FIO do FAM:= ‘Иванов’;

Пример:

Program MAIN;

Type PEREM = record

X, y: integer;

End;

Var x, y: integer;

Rec: PEREM;

Begin

X:= 10; y:= 20; { значения переменных x, y}

With rec do

Begin

X:= 3.14 \* x; { Rec.x:= 3.14 \* Rec.x }

Y:= 3.14 \* y;

End; { with }

{ если хотим, можно уточнить y:= 3.14 \* MAIN.x; }

{ MAIN – квалификатор ( имя программы или модуля UNIT) }

Пример: Распечатать все слова в файле (слово заканчивается символом ‘ ‘ или eof(f)).

Program P;

Var f: text;

Ch1, ch2: string [1];

Slovo: string [10];

Begin

Assign (f, ‘NAM.txt’);

Reset (f);

Slovo:= ‘’;

Ch1:= ‘ ‘; {пробел}

While not eof (f) do

Begin

Read (f, ch2);

If (ch2 <> ch1) then slovo:= slovo + ch2

Else begin

Writeln (slovo);

Slovo:= ‘’;

End;

End;

Writeln (slovo);

Close (f);

End.

**Процедуры:**

DELETE (var ST: string; POZ: integer; NUM: integer);

ST - входная/ выходная строка;

POZ - № позиции, с которой удаляются символы;

NUM – количество удаляемых символов.

INSERT (ST1: string; var ST: string; POZ: integer); - вставка ST1 в ST с POZ позиции.

**Процедуры преобразования числа в строку и наоборот:**

Процедура STR преобразует числовое значение в строковую переменную:

STR (X[:W:d], var ST: string);

Пример: STR (5.75: 8: 2, ST); {st= ‘5.75 ‘}

STR (5.75:8:0, ST); {st= ‘6 ‘} – округляет

Обратная процедура VAL:

Procedure VAL (st: string; var X [:W: d], var error: integer);

Error = 0, значит нормальное преобразование, без ошибок,

Error <> 0, значит ненормальное преобразование, с ошибками.

Программа на текстовый файл

Program massiv (dat, res);

Uses crt;

Const nmax = 10;

Var I, j, n, m: integer;

A: array [1 .. nmax, 1 .. nmax] of real;

Dat, res: text;

Begin

Assign (dat, ‘d: \hm.dat’);

Assign (res, ‘d: \hm.res’);

Reset (dat);

Rewrite (res);

Writeln (res, ‘ ‘, ‘matrica’);

Read (dat, n);

Readln (dat, m); { 3 4 }

For i:= 1 to n do begin {1 3 7 8 }

For j:=1 to m do {9 2 5 6 }

Read (dat, a[I, j]); {3 9 8 1 }

Readln (dat);

End;

Writeln (res, ‘chislo strok = ‘, n: 3);

Writeln (res, ‘chislo stolbcov = ‘, m: 3);

For i:= 1 to n do begin

For j:=1 to m do

Write (res, a[I, j]:7:1);

Writeln (res);

End;

Close (dat);

Close (res);

End.

**Лекция № 18**

Задание: составить программу для обработки информации о некоторых объектах – студентах какого-то института. Каждый объект характеризуется совокупностью признаков.

Для группы Пт2-97 вывести фамилии студентов мужского пола старше 18 лет.

Program p521;

Uses crt;

Type stud= record

Group, FIO, gender: string;

DOB, fiz, math, prog, count: integer;

End;

Var d, d1: stud;

Fr, fw: file of stud;

Stip, I, n: integer;

Group: string;

Begin

Clrcsr;

Write (‘Vvedite kol-vo zapicei:’);

Readln (n);

Assign (fr, ‘C:\ ved.dat’);

Assign (fw, ‘C:\ ved1.dat’);

Rewrite (fr);

For i:= 1 to n do

Begin

Clrscr;

Write (‘vvedite gruppu ‘, I, ‘-go studenta (pt2-97):’);

Readln (d. group);

Write (‘vvedite FIO ‘, I, ‘-go studenta:’);

Readln (d.fio);

Write (‘vvedite pol ‘, I, ‘-go studenta (“m” ili “zh”):’);

Readln (d.gender);

Write (‘vvedite god rozhdeniya ‘, I, ‘-go studenta:’);

Readln (d.dob);

Write (‘vvedite ocenku po fizike ‘, I, ‘-go studenta:’);

Readln(d. fiz);

Write (‘vvedite ocenku po matematike ‘, I, ‘-go studenta:’);

Readln (d. math);

Write (‘vvedite ocenku po programmirovaniu ‘, I, ‘-go studenta:’);

Readln (d. prog);

Write (‘vvedite stipendiu ‘, I, ‘-go studenta:’);

Readln (d.count);

Clrscr;

Write (fr, d);

End;

Close (fr);

Reset (fr);

Rewrite (fw);

Writeln;

Writeln (‘Studenti muzhskogo pola iz gruppi “pt2-97” starshe 18 let:’);

While not eof (fr) do

Begin

Read (fr, d1);

If (2008-d1.dob > 18) and (d1. Group = ‘pt2-97’) and (d1. Gender = ‘m’) then

Begin

Write (fw, d1);

Writeln (‘FIO:’, d1. Fio);

Writeln;

End;

End;

End.

**Множества**

- это совокупность данных, расположенных в ОП (можно выполнять логические операции). Сочетание элементов – неважно в каком порядке.

В математике {1, 2, 3}, в Pascal – [1, 2, 3]

[ ] – пустое множество; [2, 3] – подмножество.

Количество элементов множества не может быть больше 256.

256: 0, 1, 2, … , 255 (в памяти 256 бит = 32 байта).

Пример: множество целых чисел 0 : 100 : [ 5, 7, 13, 14].

[ 5, 7, 5] = [5, 7]. В памяти на месте 5 бита стоит 1, на месте 6 – 0, на месте 7 – 1 и т.д.

Описание множества:

Type <имя> = SET OF <базовый тип>;

Базовый тип – это логический, перечислимый, символьный, целый (byte).

1 способ: Var <имя1>: <имя>;

2 способ: Var <имя1>: SET of <базовый тип>;

Пример:

Type NOM = 10 .. 100;

MN = SET of NOM;

Var M: MN;

Begin

M:= [11, 15, 35];

Операции над множествами:

Отношения: [ 1, 3] = [ 3, 1] – true;

[‘A’] < [‘B’] – true

[‘A’] > [‘A’, ‘B’] – false

Объединение: + C := A + B;

Пример: Var A, B, C: set of char;

Begin

A:= [‘O’, ‘P’, ‘Q’];

B:= [‘Q’, ‘R’, ‘S’];

C:= A + B; { [‘O’, ‘P’, ‘Q’, ‘R’, ‘S’] }

C:= C + [‘X’, ‘Y’, ‘Z’];

Разность: C := A – B;

Пересечение: C := A \* B;

Функция in определяет, входит ли элемент (выражение) во множество или нет.

If ‘o’ in C then …

Задача: является ли некоторое значение буквой?

1 способ: Var c: char;

Begin

C:= readkey;

If (c >= ‘A’) and (c <= ‘Z’) then writeln (c, ‘ – буква’);

2 способ: if c in [‘A’..’Z’] then writeln (c, ‘ – буква’);

Процедуры:

INCLUDE (S, I) – включить элемент I во множество S.

EXCLUDE (S, I) – исключить элемент I из множества S.

Эти процедуры выполняются быстрее, чем операции “+” и “-“.

Ввод множества

Такого оператора в Pascal нет. Поэтому нужно использовать следующий цикл:

Var A: set of 1..100;

K, i: integer;

Begin

A:= [];

For k:= 1 to 25 do

Begin

Readln(i);

A:= A + [i];

End;

End.

Команды печати множества тоже нет. Поэтому используют тоже цикл:

For i:=1 to 100 do

If I in A then writeln (i);

Задача: Нахождение простых чисел

Z = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, …, 100]

P = [1, 2, 3, 5, 7, 11, 13, 17, 19, …] – простые числа

{ Вирт: Решето Эратосфена }

Program P1;

Type S = set of 1..100;

Var Z, P: S;

I, k: integer;

Begin

Z:= [2..100]; P:= [1]; i:= 2;

Repeat

While not (I in Z) do i:= i+1;

P:= P + [i];

K:= I;

While (k <= 100) do

Begin

Z:= z – [k];

K:= k + i:

End;

Until Z:= [ ];

For i:= 1 to 100 do

If I in P then write (i:4);

Writeln;

End.

Задача: отсортировать массив (по возрастанию).

Для этого нужно ввести массив чисел и сформировать из него множество (повторяющиеся элементы будут удалены).

Program Pr;

Type S = set of 1..100;

Var Z: S; I, k: integer;

M: array [1..10] of 1..100;

Begin

Z:= [ ];

For i:= 1 to 10 do

Begin

Readln(M[i]);

Z:= z + [M[i]];

End;

For i:= 1 to 100 do

If I in Z then write (i:4);

Writeln;

End.

Задача: C = A – B

Procedure MINUS (A, B: mas; n, m: integer; Var C: mas; var k: integer);

Var I, j: integer; F: Boolean;

Begin

K:= 0;

For i:= 1 to n do

Begin

F:= false;

For j:= 1 to m do

Begin

If (A[i] = B[j]) then

Begin

F:= true;

Break;

End;

End;

If not F then begin

K:= k+1;

C[k]:= A[i];

End;

End;

End;

Задача Множества

Program Mn1;

Uses crt;

Type simv = ‘A’..’Z’;

Mn = Set of simv;

Var MnA, MnB, MnC, MnD: Mn;

Ch: char;

Procedure VvodMn (Var MnX: Mn; Name: char);

Var ch: char;

Begin

MnX:= [ ];

Write (‘Ввод множества ‘, Name, ‘ >’);

Ch:= Readkey;

While (Ord (ch) <> 13) do begin

If (ch in [‘A’..’Z’]) and not (ch in MnX) then begin

MnX:= MnX + [ch];

Write (ch);

End;

Ch:= Readkey;

End;

Writeln;

End;

Procedure VivodMn;

Var I: simv;

Begin

Writeln;

Write (‘Множество A>’);

For i:= ‘A’ to ‘Z’ do

If (I in MnA) then write (i);

Writeln;

Write (‘Множество B>’);

For i:= ‘A’ to ‘Z’ do

If (I in MnB) then write (i);

Writeln;

Write (‘Множество C>’);

For i:= ‘A’ to ‘Z’ do

If (I in MnC) then write (i);

Writeln;

Write (‘Множество D>’);

For i:= ‘A’ to ‘Z’ do

If (I in MnD) then write (i);

Writeln;

End;

Begin

Clrscr;

VvodMn (MnA, ‘A’);

VvodMn (MnB, ‘B’);

VvodMn (MnC, ‘C’);

MnD:= [ ];

VivodMn;

Writeln (‘A – B > D’);

MnD:= MnA – MnB;

VivodMn;

Writeln (‘MnD \* MnC > Mn’);

MnD:= MnD \* MnC;

VivodMn;

Ch:= Readkey;

While (Ord (ch) <> 13) do begin

If (ch in [‘A’..’Z’]) then begin

If (ch in MnD) then writeln (ch, ‘ принадлежит MnD’)

Else writeln (ch, ‘ не принадлежит MnD’);

Ch:= readkey;

End;

End;

End.