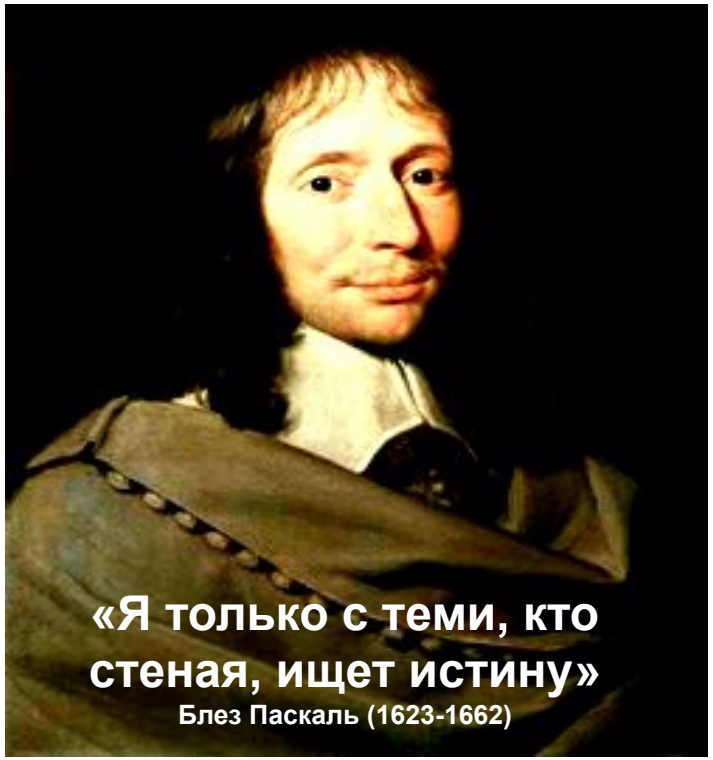
2010



PASCAL

ВВЕДЕНИЕ В ОСНОВЫ ПРОГРАММИРОВАНИЯ

Цели курса:

- систематизация знаний по основным разделам курса;
- знакомство с полиграфическими, методическими материалами Образовательного центра «Школьный университет»;
- обмен опытом между преподавателями.

Концепция курса:

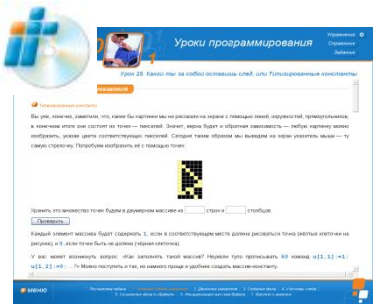
- знакомство с тематическим курсом «Основы алгоритмизации и программирования на языке Pascal»;
- поэтапное изучение всех разделов данного курса с практическим закреплением теоретической части курса;
- использование раздаточного материала, выполнение практических работ, тестов.

Методический комплект

Учебный материал

Методический материал

Учебно-методическое пособие (курс 70 часов)
Электронный практикум



Глава 1. Алгоритмизация
Глава 2. Знакомство с Pascal
Глава 3. Основные алгоритмические конструкции
Глава 4. Структурированные типы данных

Лекция 1.
Курс «Основы алгоритмизации и программирование на языке Pascal» — новый подход

Контрольно-измерительный пакет:

- ***набор контрольных работ;***
- ***ведомости оценок.***

Программы, методические рекомендации:

- ***цели и задачи изучения курса;***
- ***рабочие программы;***
- ***тематический план;***
- ***содержание.***

На диске учителя

Концепция курса

Глава 1. Алгоритмизация
Глава 2. Знакомство с Pascal
Глава 3. Основные алгоритмические конструкции
Глава 4. Структурированные типы данных

Алгоритмизация

Среда исполнителя Крошка Ру

Программирование на языке Pascal

Установка Borland Pascal и Free Pascal

Лекция 2.
Исполнитель Крошка Ру и алгоритмизация

Лекция 3.
Программное обеспечение: где, как, сколько?

Рабочая программа КПК (Turbo Pascal)

Раздел 1. Знакомство с Pascal		**5 часов**
Тема 1	Интегрированная среда разработки программ на языке Pascal. Основы языка. Типы данных. Процедуры ввода и вывода.	1 час
Тема 2	Модули и подпрограммы.	2 часа
Тема 3	Графический модуль.	2 часа
Раздел 2. Основные алгоритмические конструкции		**7 часов**
Тема 4	Оператор ветвления.	2 часа
Тема 5	Оператор повтора.	2 часа
Тема 6	Символьный тип данных.	2 часа
Тема 7	Графика. Анимация.	1 час
Раздел 3. Структурированные типы		**8 часов**
Тема 8	Массивы.	2 часа
Тема 9	Строки.	1 час
Тема 10	Работа с файлами.	2 часа
Тема 11	Множества и записи.	2 часа
Резерв		1 час

20 часов

Основные этапы решения задач на ЭВМ

Первый этап. **Постановка задачи.**

Второй этап. **Математическое или информационное моделирование.**

Третий этап. **Алгоритмизация задачи.**

Свойства алгоритма:

1. Понятность.
2. Дискретность.
3. Определённость.
4. Результативность.
5. Массовость.

Обзор урока ЭП :

Урок 4 «Независимое расследование, или Что же такое алгоритм».

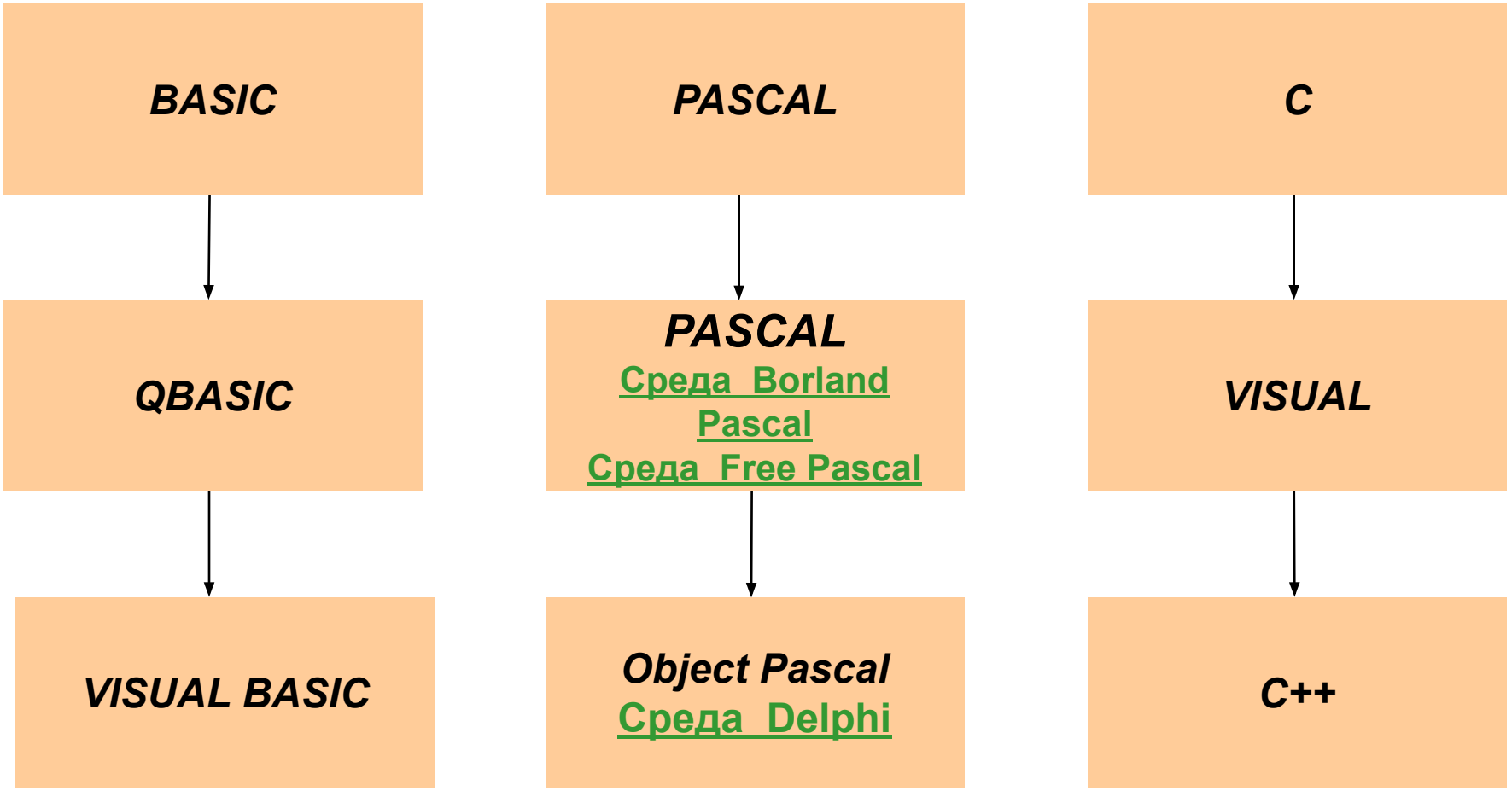
Четвёртый этап. **Программирование.**

Пятый этап. **Ввод программы и исходных данных в ЭВМ.**

Шестой этап. **Тестирование и отладка программы.**

Седьмой этап. **Исполнение отлаженной программы и анализ результатов.**

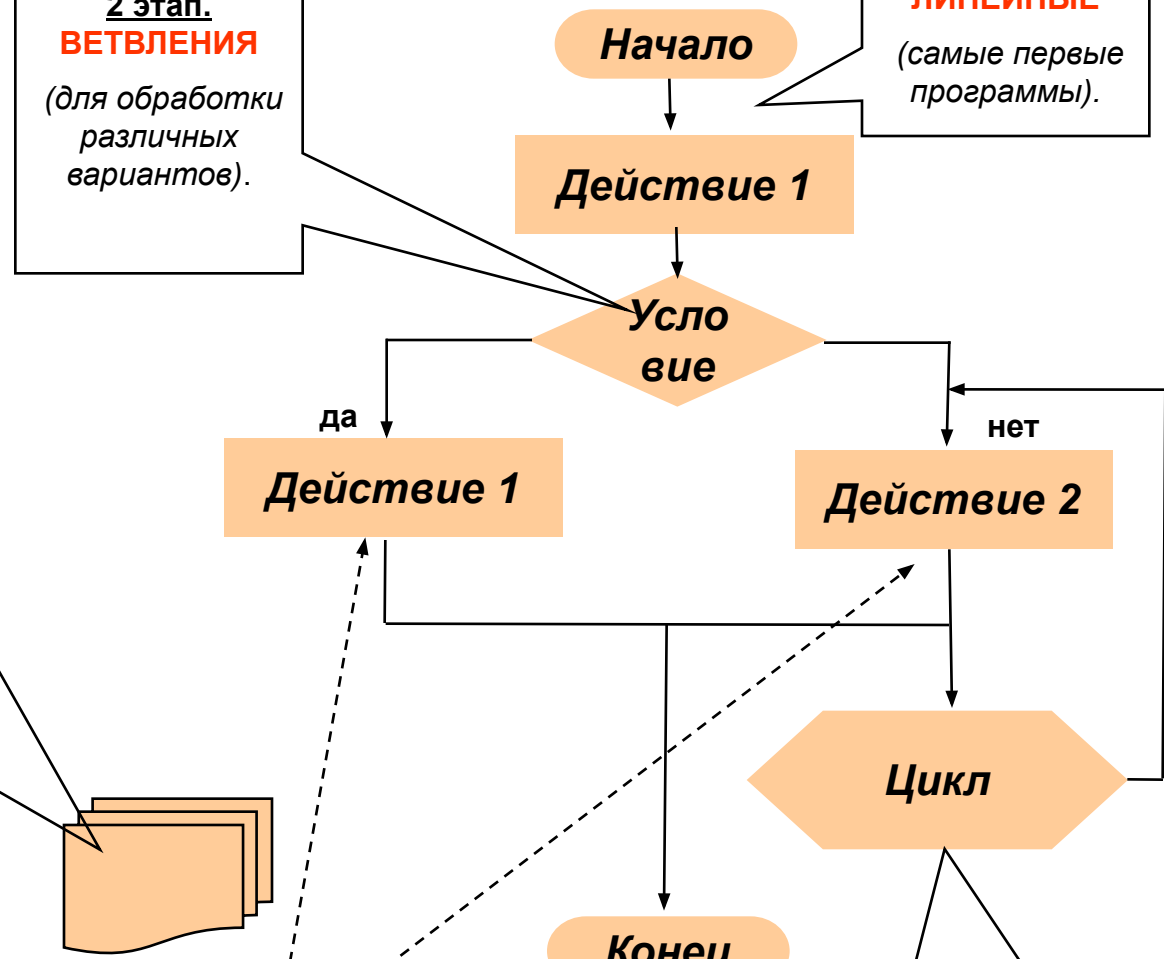
Развитие семейств языков



Эволюция программирования

1 этап.
ЛИНЕЙНЫЕ
(самые первые программы).

2 этап.
ВЕТВЛЕНИЯ
(для обработки различных вариантов).



5 этап. **(БИБЛИОТЕКИ) МОДУЛИ**
(подпрограммы занимают много места в файле программы и их образовали в отдельные файлы в машинных кодах).

6 этап. **ОБЪЕКТЫ**
(в программах встречались похожие части , отличающиеся в деталях).

7 этап. **КЛАССЫ ОБЪЕКТОВ С ВОЗМОЖНОСТЬЮ НАСЛЕДОВАНИЯ.**

4 этап. **ПОДПРОГРАММЫ**
(многие действия выполнялись в различных местах программы).

3 этап. **ЦИКЛЫ**
(для выполнения многократных действий).

Основные достоинства языка Pascal

1. Отвечает требованиям структурного программирования

Позволяет строить программу из отдельных блоков.

- *применяются три управляющие конструкции: следование, выбор, повторение;*
- *структура программы отражает структуру данных;*
- *на первом этапе проводится проектирование программы, а на втором её написание.*

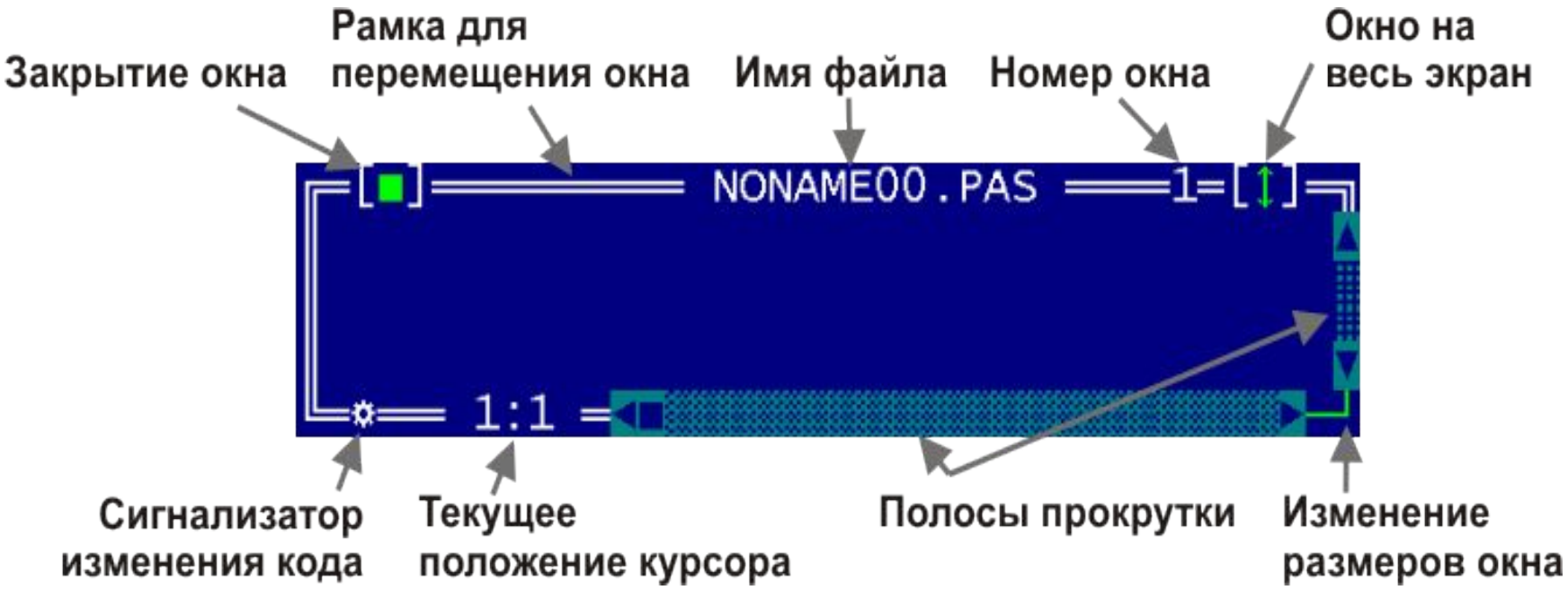
2. Строго типизированный язык

Содержит полный набор структурных типов данных, а также развитые средства построения из них новых типов данных.

Интерфейс среды Pascal

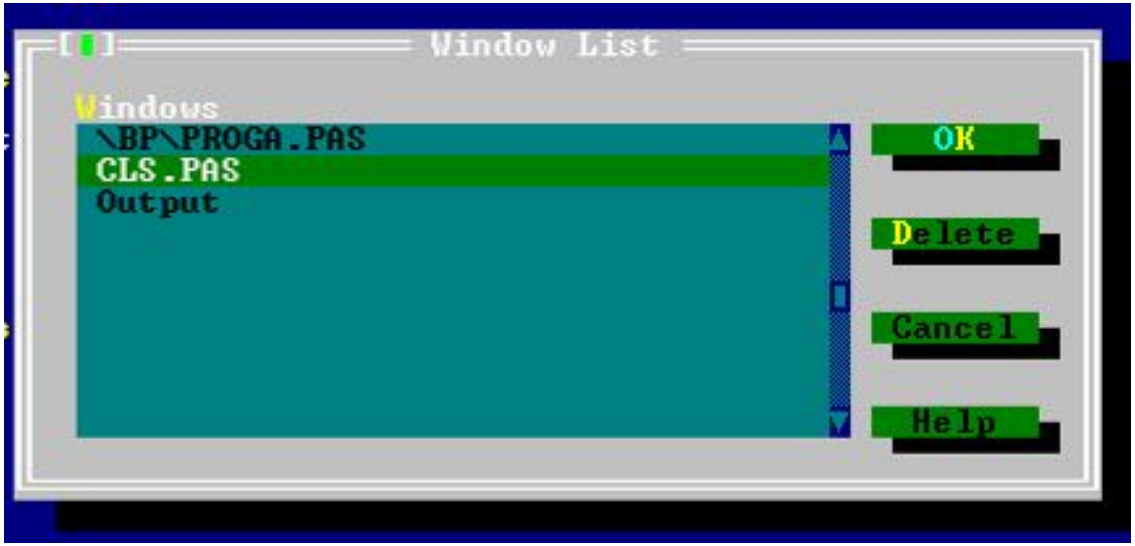
Интерфейс включает в себя:

- многооконный текстовый редактор;
- компилятор, компоновщик программ;
- отладчик программ;
- систему помощи.



Интерфейс среды Pascal

Отображение результатов выполнения программы



Список открытых файлов: Alt + 0.

Быстрый доступ к открытым файлам: Alt + <№ окна>.

Просмотр текущего значения переменных: Ctrl + F7.

Для выполнения программы по строкам: F8.

Компиляция + проверка: F9.
Запуск: Ctrl+F9.

Работа с данными

ОПРЕДЕЛЕНИЕ ТИПА ДАННЫХ

Однозначность операций над данными

ИДЕНТИФИКАЦИЯ ПЕРЕМЕННЫХ

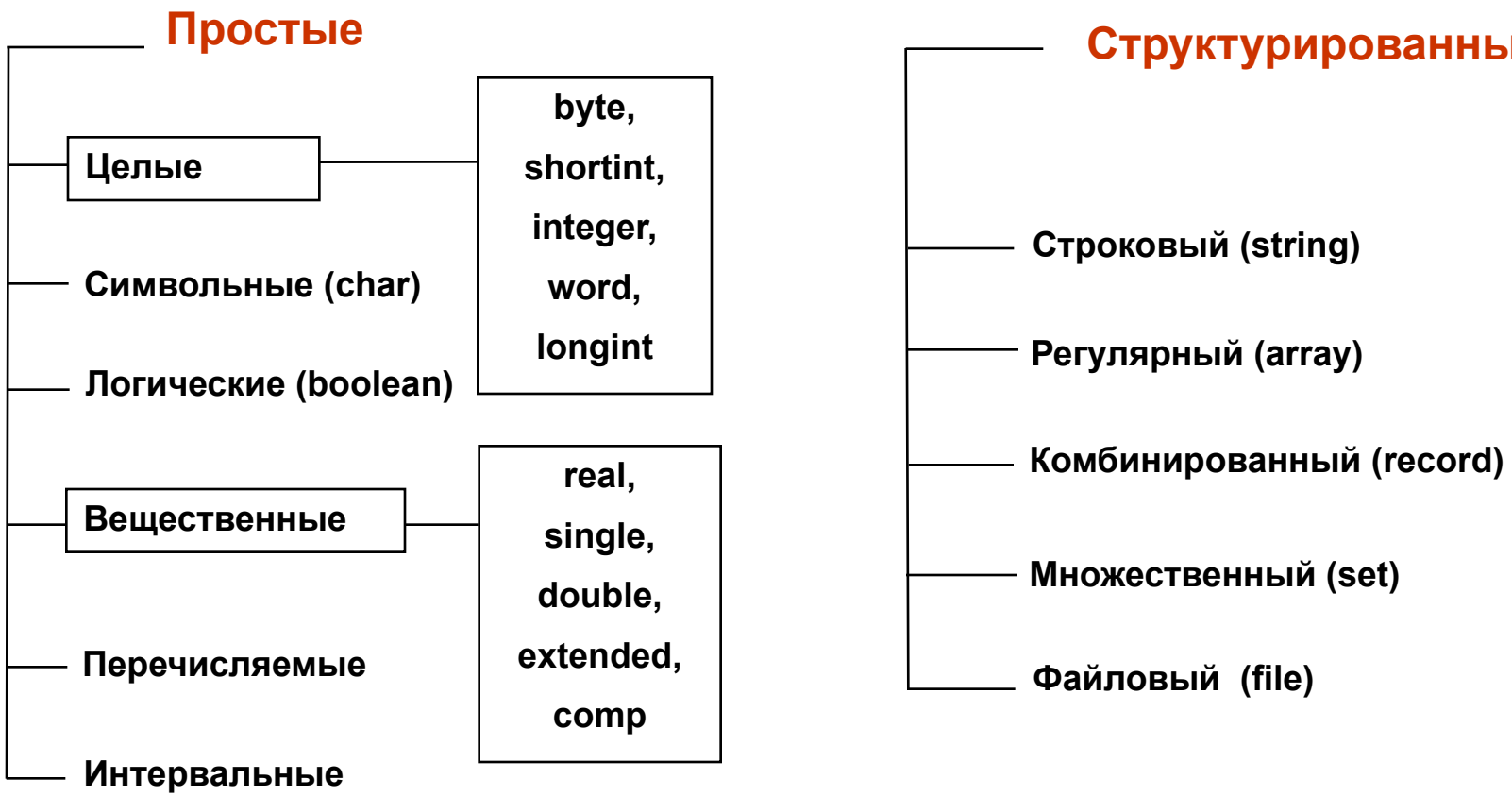
Объявление идентификаторов

ЗАДАНИЕ ЗНАЧЕНИЙ

Операции с данными

Типы данных

Тип данных — это характеристика идентификатора, определяющая множество значений, которые он может принимать (целые или дробные числа, строки и т. д.).



Простые типы: одна переменная — одно значение.
Структурированные типы: одна переменная — несколько значений.

Целый и вещественный типы данных

Конечный набор возможных значений

Тип	Диапазон значений
byte	0…255
shortint	− 128…127
word	0…65535
integer	− 32768…32767
longint	− 2147483648…2147483647

Тип	Диапазон десятичного порядка
real	−39…+38
single	−45…+45
double	−324…+308
extended	−4932…+4932
comp	−263+1…263 −1

Выход за пределы диапазона приводит к ошибке

Синтаксис:

```
Var <имя переменной>:<тип переменной>;
```

Резервирует место в памяти компьютера под переменные: **a**, **x**, **y**.

```
Program My_program;
uses CRT; {подключение модуля CRT}

var
   a:integer;
   x,y:real;
begin

ClrScr;{процедура очистки экрана}


Readkey;{ожидание нажатия клавиши}

end.
```

Операции с целыми переменными

Арифметические операции: `Sqr ,+, –, *, /`

Нельзя использовать с целыми типами

Стандартные функции:

div — вычисляет целую часть от частного, дробная откидывается.	`10 div 3=3;` `2 div 3=0;`
mod — вычисляет остаток, полученный при делении.	`11 mod 5 = 1;` `14 mod 5 = 4;`

```
Program My_program;
uses CRT;   {подключение модуля CRT}
var
   x,y:integer;

begin
ClrScr;  {очистка экрана}
writeln('введите два числа');
write('x='); read(x);
write('y='); read(y);
writeln(x ,' div ',y,'=',x div y );
writeln(x ,' mod ',y,'=',x mod y );
readkey;  {ожидание нажатия клавиши}
END.
```

Работа функций используется в операторе вывода.

```
введите два числа
x=10
y=3
10 div 3=3
10 mod 3=1
```

Операции с вещественными переменными

Арифметические операции: `Sqr`, `+`, `-`, `*`, `/`

Стандартные функции: `Pi`, `Sqrt`, `Sin`, `Cos`, `Abs`, `Exp`, `Ln`.

вещественный → вещественный:

`Frac`, **`Int`**;

вещественный → целый:

`Round`, **`Trunc`**.

- ✔ вычисление дробной части числа **`Frac(5.67)=0.67`**
- ✔ вычисление целой части числа **`Int(5.67)=5.0E+00`**
- ✔ округление вещественного числа до ближайшего целого **`Round(5.67)=6`**
- ✔ отбрасывание дробной части числа **`Trunc(5.67)=5`**

```
uses crt;     {подключение модуля CRT}
var x:real;
begin
ClrScr; {очистка экрана}
writeln('введите любое дробное число');
write ('x='); read(x);

writeln('Frac(' ,x, ')=', frac(x),' форматируем и получаем - ', frac(x):6:2);
writeln('Int(' ,x, ')=', int(x),' форматируем и получаем - ', int(x):2:0);
writeln('Round(' ,x, ')=', round(x));
writeln('Trunc(' ,x, ')=', trunc(x));
readkey;
end.
```

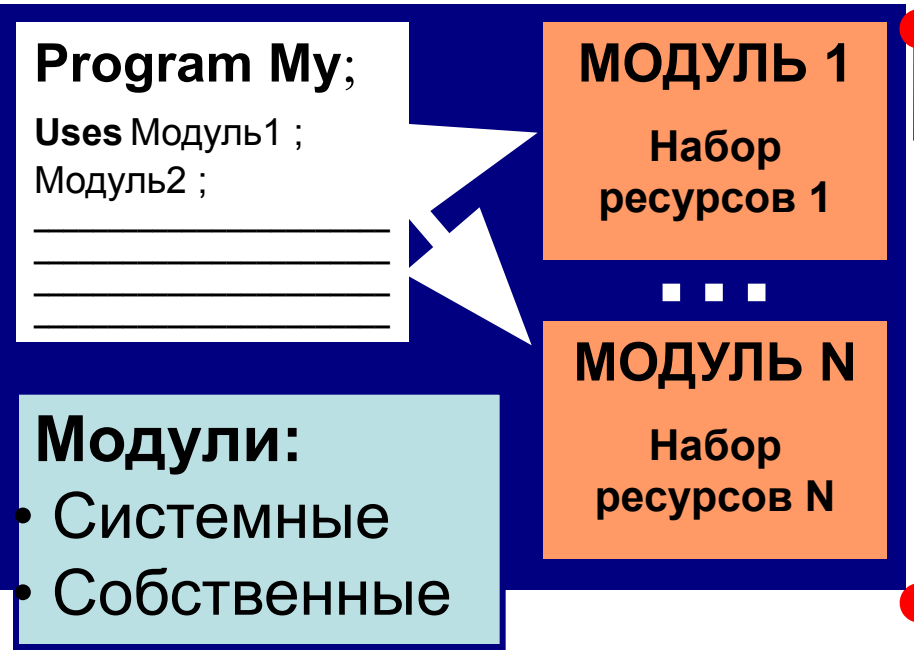
```
Output
введите любое дробное число
x=5.67
Frac( 5.6700000000E+00)= 6.7000000000E-01 форматируем и получаем -   0.67
Int( 5.6700000000E+00)= 5.0000000000E+00 форматируем и получаем -  5
Round( 5.6700000000E+00)=6
Trunc( 5.6700000000E+00)=5
```

Подключение дополнительных модулей

Модуль расширяет возможности программ путём введения дополнительных операторов, стандартных процедур и функций.

Пример включения стандартных модулей:

```
uses crt, dos, graph, printer.
```



Подключённый модуль с именем **CRT**.

```
Program My_program;
uses CRT; {подключение модуля CRT}

var
    a:integer;
    x,y:real;
begin

ClrScr;{процедура очистки экрана}


Readkey;{ожидание нажатия клавиши}

end.
```

Очистка текстового экрана.

Ожидание нажатия на клавишу.

Запрос данных с клавиатуры

Ввод информации с клавиатуры обеспечивает процедура ввода:
Read или **ReadLn.**

Синтаксис:

Read (N1, N2, … Nn);

Где **N1, N2, … Nn** — переменные (целые, вещественные, строковые).

Read(Ln) — курсор устанавливается на следующую строку.

В переменную ***X*** и ***A*** заносится значение, введённое с клавиатуры.

```
Program My_program;
uses CRT;   {подключение модуля CRT}
var
   x,y:real;
   a:integer;
begin
ClrScr;  {очистка экрана}
writeln('введите два числа');
write('x='); read(x);
write('a='); read(a);
readkey;  {ожидание нажатия клавиши}
END.
```

- После ввода значения, необходи-мо нажать клавишу **Enter**.
- Если переменных в операторе указано несколько, то они вводятся через **пробел**, либо через нажатия клавиши **Enter**.

```
введите два числа
x=2.5
a=78
```

Ввод данных

Ввод данных с клавиатуры в текстовом режиме:

1. Через функцию **ReadKey** для чтения первого байта из очереди нажатий на клавишу.

2. Через процедуру ввода **Read(Ln)**

Ввод данных с клавиатуры непосредственно в программе:

3. Через оператор присваивания **:=**.

```
uses crt;
var c:char;
begin
ClrScr;
writeln('нажмите  клавишу') ;
c:= readkey; {ожидания нажатия
пользователем клавиши и занесение
символа, соответствующего нажатой
клавише, в переменную C}
write('c=' ,c);{ ожидание нажатия
пользователем клавиши}
readkey;
end.
```

```
3:20
Output -2
нажмите  клавишу
c=s
```

Тип переменной должен совпадать с **типом вводимых значений** для этой переменной.

Операторы присваивания

Для задания значения переменной необходимо воспользоваться оператором присваивания **:=**

Синтаксис:

```
<Переменная>:=<Значение>;
```

```
Program My_program;

Var A:Integer;
    X,Y:Longint;

Begin

A := 3;

End.
```

В переменную (целочисленную) с именем A заносится значение 3.

Вывод информации на экран

Вывод информации на монитор обеспечивает процедура вывода: **Write** или **WriteLn**.

Синтаксис:

```
Write (N1, N2, … Nn);
```

N1, **N2**, … **Nn** — переменные (целые, вещественные, строковые).

WriteLn — перемещает курсор на следующую строку.

```
Program My_Program;

Begin

 Write('Message 1');
 Write('Message 2');
 Write('Message 3');

End.
```

```
Message 1Message 2Message 3
```

```
Program My_program;
uses CRT;   {подключение модуля CRT}
var
   x,y:real;
   a:integer;
begin
ClrScr;  {очистка экрана}
writeln('введите два числа');
write('x='); read(x);
write('a='); read(a);
write(x);
writeln;
write(a);
readkey;  {ожидание нажатия клавиши}
END.
```

```
введите два числа
x=4
a=56
 4.0000000000E+00
56
```

«Пустой» оператор **WriteLn** добавляет пустую строку.

Форматы вывода

Значение	Выражение	Результат
5671	`Write(I,I);` (вывод на экран содержимого ячейки `I` два раза)	56715671
134	`Write(I:6);` (выводит значение `I` в крайние правые позиции полей шириной равной 6)	- - -134
312	`Write(I+I:3);` (содержимое `I` удваивается, и результат выводится с 3 позиции)	624
7,154E+2	`Write(I:6:1);` (6 — задаёт количество позиций под всё число, включая фиксированную точку; 1 — задаёт количество позиций под дробную часть числа)	-715.4
	`Write('Сумма=');` (текст обрамляется апострофами)	Сумма =

Вывод в произвольное место экрана

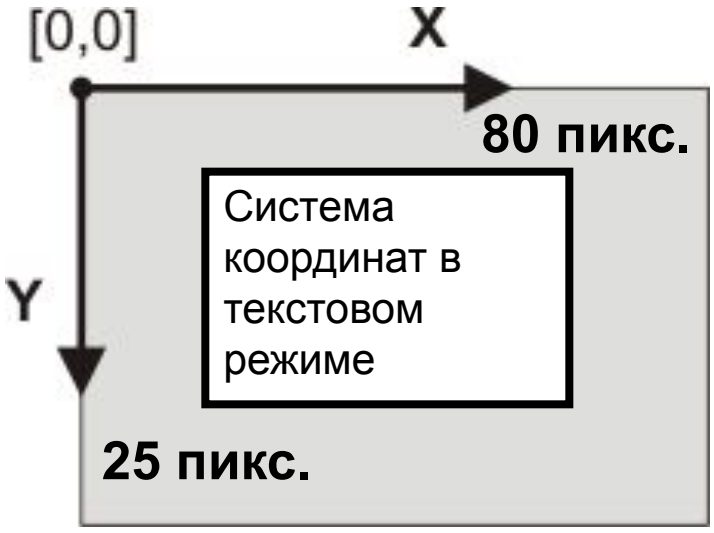
Синтаксис:

```
GotoXY(X,Y:Integer);
```

X, Y — координата знако-места на экране.

```
Program My_program;
{Подключение модуля}
Uses Crt;
Begin
 {Очистка экрана}
 ClrScr;
 {Вывод данных}
 GotoXY(1,  1); write('▒');
 GotoXY(80, 1); write('▒');
 GotoXY(1,  25); write('▒');
 GotoXY(80, 25); write('▒');
 {Задержка экрана}
 ReadKey;
End.
```

Программа выводит по углам экрана символ «▒» (ASCII-код 177).



Вывод в цвете

TextColor(Color);

Определяет цвет символов.

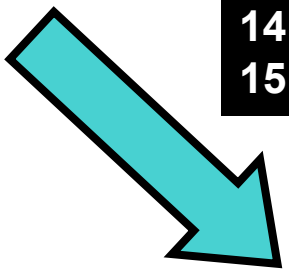
TextBackground(Color);

Определяет цвет знако-места.

```
Program MyProgram;
Uses Crt;
Begin
 TextColor(Red);
 TextBackGround(Blue);
 Write('На дворе ');
 TextColor(White);
 Write('трава, ');
 TextColor(Green);
 TextBackGround(Yellow);
 Write('на траве ');
 TextBackGround(Magenta);
 Write('дрова.');
End.
```

Константы цвета модуля CRT

0	Black – чёрный
1	Blue – синий
2	Green – зелёный
3	Cyan – циановый
4	Red – красный
5	Magenta – сиреневый
6	Brown – коричневый
7	LightGray – светло-серый
8	DarkGray – тёмно-серый
9	LightBlue – голубой
10	LightGreen – светло-зелёный
11	LightCyan – светло-циановый
12	LightRed – розовый
13	LightMagenta – светло-сиреневый
14	Yellow – жёлтый
15	White – белый



Отладка

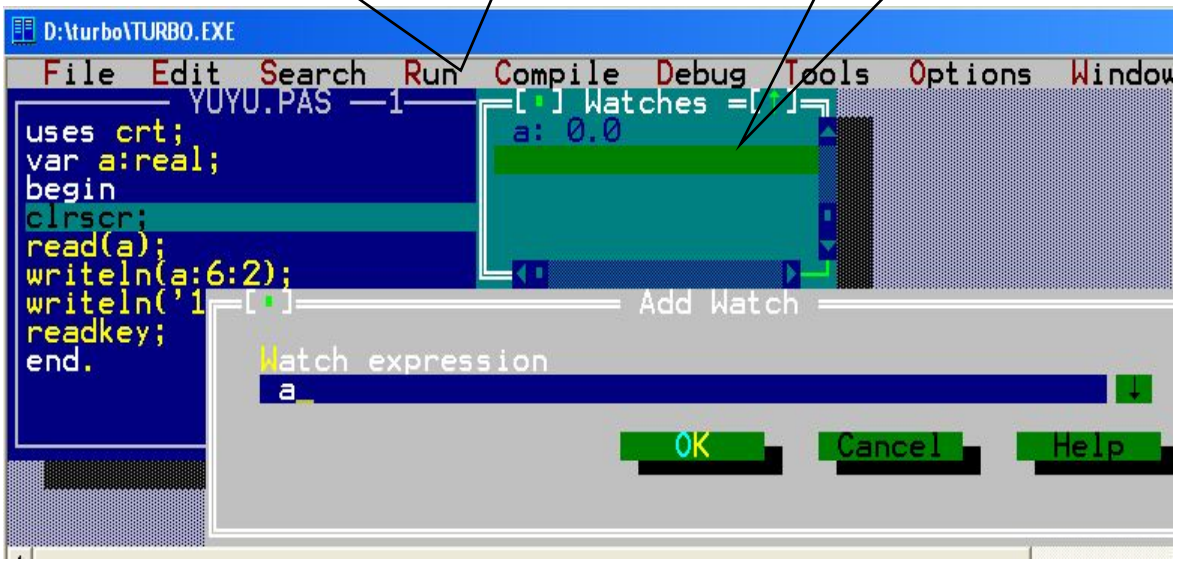
Синтаксические — несоблюдение правил языка, исправляются на стадии компиляции.

Семантические — завися от конкретных значений переменных, возникают на стадии выполнения программы.

Логические — ошибки в алгоритме: программа работает так как написана, но не так как требуется.

- компиляция программы: **Run-Run**;
- пошаговый режим отладки: **F7**;
- задание значений переменных на просмотр: **Ctrl+F7**.

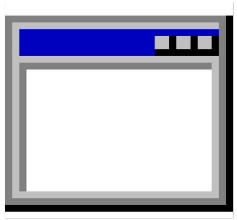
Окно просмотра значений переменных



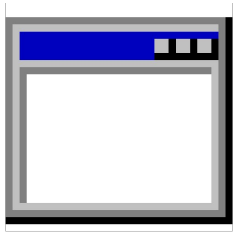
Обзор материала ЭП :

Прогон и отладка

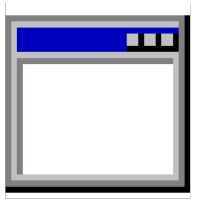
ПРИМЕР РАБОТЫ ПРОГРАММЫ



Color.exe



Delay.exe



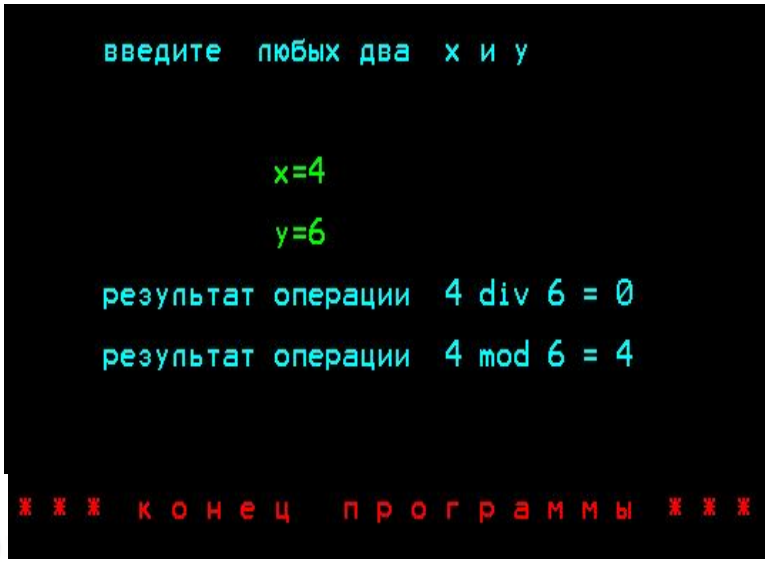
ÑÀÌÎË¨Ò.EXE

ЗАДАНИЕ

1. **Выполните программу** вычисления над двумя числами ***x*** и ***y***, используя различные функции и дополнив вывод результата через функции управления координатами и цветом.

2. **Выполните программу** нахождения с помощью линейного алгоритма наибольшего из двух заданных A и B.

3. **Обзор материала ЭП.** «Метод дихотомии».



Переменные объявляются в разделе **Var**.

Целый тип называется **Integer**.

Вещественный тип называется **Real**.

Операторы ввода вывода: **Read(Ln)**, **Write(Ln)**.

Синтаксис присвоения переменной значения: **<Переменная>:=<Значение>;**

После каждого оператора ставится знак **;** (кроме **begin** и последнего **end**).

- **GotoXY(X,Y:Integer)** — координата знако-места на экране.
- **TextBackground(Color)** — определяет цвет знако-места.
- **TextColor(Color)** — определяет цвет символов.

Описание подпрограмм

Подпрограмма — часть программы, оформленная в виде отдельной синтаксической конструкции и снабжённая именем (самостоятельный программный блок), для решения отдельных задач.

Процедуры

Описание процедуры:

```
procedure<имя> (<список формальных параметров>)
  {раздел выполнения локальных имён}
Begin
  {раздел выполнения операторов}
End;
```

Вызов процедуры:

```
<имя >(<список фактических переменных>);
```

Функции

Описание функции:

```
function<имя> (<список формальных параметров>): тип;
  {раздел описания локальных имён}
Begin
  {раздел выполняемых операторов}
<Имя функции>:=<значение>;
{обязательный параметр}
End;
```

Вызов функции:

```
< оператор>:= <имя функции>
 (<список фактических  переменных>);
```

1. В правой части оператора присваивания.
2. В выражении, стоящем в условии оператора разветвления.
3. В процедуре вывода, как результат работы функции.

Понятие глобальных и локальных переменных

Глобальные имена действуют в пределах нескольких вложенных блоков.
Локальные имена действуют в пределах одного блока.

Механизм передачи данных

Список параметров предназначен для обмена информацией между вызывающей и вызываемой подпрограммами.

Параметры:
- **формальные**
- **фактические**

Согласованы по:

количеству, месту,

типу и способу передачи

Подпрограмма
Программа

По способу взаимодействия вызывающей и вызываемой подпрограмм параметры подразделяются на:

- **входные**
- **выходные**
- **входные и выходные**

Для обеспечения такого взаимодействия используются:

- **параметрами-значениями**
- **параметрами-переменными**
- **параметрами-константами**

Механизм передачи данных

Параметры-значения — **механизм передачи по значению**

Используются только для входных параметров.

Параметры-переменные — **механизм передачи по адресу**

Используется для выходных параметров, а также входных и выходных параметров.

Параметры-константы — **механизм по адресу**

В ходе выполнения процедуры значение формального параметра изменять нельзя.

Параметры-константы экономят память, так как под адрес выделяется всего четыре байта.

```
Procedure MyProc(Par1:integer; Var Par2:real; const Par3:Type3);
```

ЗАДАНИЕ

1. **Выполните программу**, которая вычисляет расстояние между тремя точками с помощью подпрограмм.

2. **Проверьте себя**: Задания к уроку 8. «И снова уравнение, или Подпрограммы» электронного практикума.

Задание 4

Задание 5

Используемый материал:

Procedure *<имя> <список входных переменных>* ; **Var** *<список выходных переменных >*;

Function *<имя> <список входных переменных: тип> : тип;*

Графика

Для работы с графикой в Pascal необходимы два файла:

- модуль **graph.tpu** (*находится в каталоге в \BGI).*
- драйвер графического адаптера **egavga.bgi** (*находится в каталоге \UNITS).*

Чтобы рисовать, надо:

1. Подключить модуль **GRAPH** (в разделе **Uses**).
2. Инициализировать графику (**InitGraph**).
3. Что-нибудь нарисовать.
4. Закрыть графический режим (**CloseGraph**).

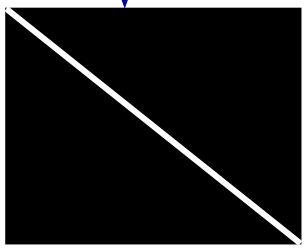
Инициализация графика


```
PROGRAM  Primer_1;
Uses Graph, crt;
Var  Gd,Gm: Integer;
BEGIN
   Gd:=0;
   InitGraph (Gd,Gm,'d:\BP\bgi');
   Line (0,0,639,479);
   ReadKey;
   CloseGraph;
END.
```

Инициализация графического режима

Рисование линии. Перо переходит из точки (0,0) в точку (639, 479).

Закрытие графического режима



Заливка объектов

`SetFillStyle (x,y);` — устанавливает маску заливки и цвет фона.

`FloodFill (x,y,z);` — устанавливает координаты заливки.

Если указаны координаты внутри фигуры — заливка фигуры.

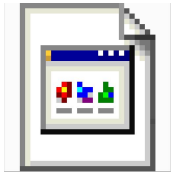
Если указаны координаты вне фигуры — заливка фона.

Используемый материал:

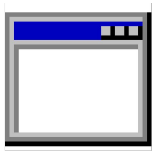
- графический модуль: `Graph`;
- инициализация графики: `InitGraph`;
- закрытие графического режима: `CloseGraph`.

ПРИМЕР РАБОТЫ ПРОГРАММЫ

ÃÈÑÒÎÃÐÀ.EXE



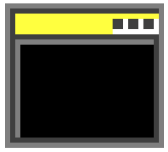
ÊÀÔÅËÜ.EXE



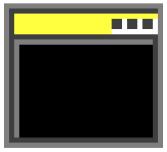
ØËßÏÀ.EXE

Практическое занятие

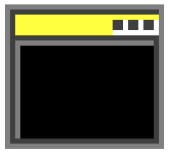
1. **Проверьте себя:** Задания к уроку 10 «Белокрылые лошадки, или Относительные координаты» электронного практикума.



Задание № 1



Задание № 4



Задание № 5

2. **Выполните задание** на выбор урока 10 электронного практикума.

Задание № 11

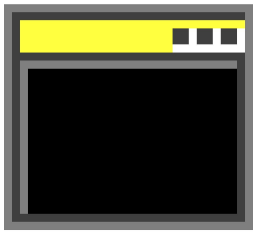
Задание № 12

Задание № 14

Задание № 16

Контрольный обзор по разделу

Раздел 1. Знакомство с Pascal		5 часов
Тема 1	Интегрированная среда разработки программ на языке Pascal. Основы языка. Типы данных. Процедуры ввода и вывода	1 час
Тема 2	Модули и подпрограммы	2 часа
Тема 3	Графический модуль	2 часа



ТЕСТ
Раздел 1

1. **Проверьте себя:** Задание к уроку 6 «Посчитаем, или Типы данных» электронного практикума.

Задание № 8

2. Проверьте себя: Задание к уроку 8 «И снова уравнение, или Подпрограммы» электронного практикума.

Задание № 5

Операторы ветвления

В языке Pascal имеются два оператора ветвления:

- **`If`** (ветвление по условию);
- **`Case`** (ветвление по выбору).

Если **число уровней вложения условного оператора `If` больше двух-трёх**, то лучше воспользоваться оператором ветвления по **выбору `Case`**.

Условный оператор If

Условный оператор **If** реализует «ветвление», изменяя порядок выполнения операторов в зависимости от истинности или ложности некоторого условия.

Краткая форма (К. Ф.) :

```
If <условие> then <оператор>;
```

```
If X<0 Then X := 1;
```

Полная форма (П. Ф.):

```
If <условие> then <оператор_1>
   else <оператор_2>;
```

После слов **Then** и **Else** можно использовать только один оператор.

Например:

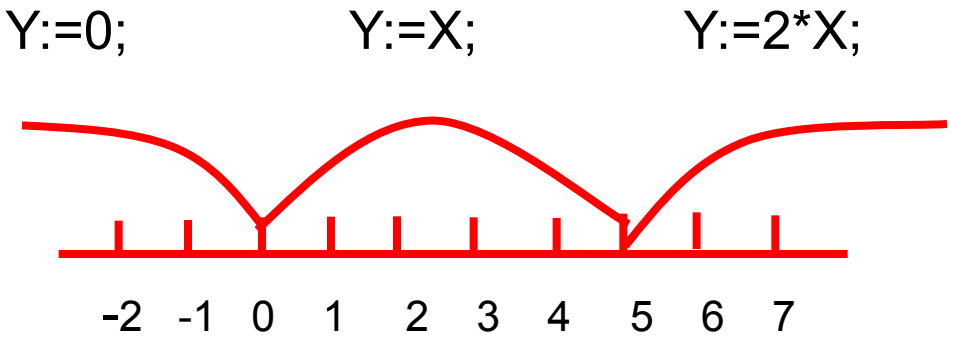
```
If X>5 Then X := X - 1
       Else X := X + 1;
```

Перед словом **Else**, знак **;** отсутствует.

Вложенная структура условного оператора If

Пример 1

$$y = \begin{cases} 0, & если \quad x < 0 \\ x, & если \quad 0 \le x < 5 \\ 2*x, & если \quad x \ge 5 \end{cases}$$



1. Объединить в II крупных варианта.

x<0 **x>=0**

```
if  x<0 then y:=0  else
```

2. Разделить II-ой вариант на 2 группы.

x<5 **x>=5**

```
if x<5 then y:=x  else  y:=2*x;
```

П. Ф. внешнее условие

```
if x<0 then y:=0
  else
     if x<5 then y:=x
        else y:=2*x ;
```

П. Ф. вложенное условие

Вложенная структура условного оператора If

Пример 2

Значение ***a,b,c*** — целые числа. Определить наибольшее и занести в переменную ***max***.

1 способ

П. Ф.

```
if a>b then max:=a
   else max:=b;
if max<c then max:=c;
```

К. Ф.

Надо помнить!!!

Условный оператор можно вставить:

1. После слова **`else`**;

2. После слова **`then`**;

2 способ

П. Ф.

внешнее условие

П.Ф. вложенное условие

```
read(a,b,c);
if a>b then
  if a>c then max:=a
    else max:=c
else
    if b>c then max:=b
       else max:=c;
```

П.Ф. вложенное условие

Проверка условия

Операции сравнения:

`>` — больше;
`<` — меньше;
`=` — равно;
`>=` — больше или равно;
`<=` — меньше или равно;
`<>` — не равно.

Мультипликативные операции:

`div` — целая часть от деления;
`mod` — остаток от деления.

Логические операции:

`not` — **Не**;
`and` — **И**;
`or` — **Или**.

Вычислить: y = 16,5x + 9x – 12,5x , при x = [-5..-1, 1..5]

```
((x>=-5) and (x<=-1)) OR ((x>=1) and (x<=5));
```

При использовании логических операций условия заключаются в скобки

Операторные скобки

Если после слов **Then** или **Else** необходимо записать несколько операторов, то их заключают в ***операторные скобки (составной оператор)***.
Операторные скобки начинаются словом **Begin**, а заканчиваются словом **End**.

Например:

```
If Z > 0 Then Begin
   X := 1;
   Y := -1;
   WriteLn( 'Информация принята' );
End
Else
   WriteLn( 'Ошибка' );
```

ЗАДАНИЕ

1. **Выполните программу:** Определите чётность либо нечётность вводимого числа функцией **mod**.
2. **Выполните программу:** Напечатайте фразу: «Мы нашли в лесу _ грибов». Согласуйте окончание слова «гриб» с введённым числом (количество грибов от 1 до 30 вводится с клавиатуры).

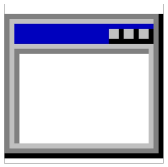
Используемый материал:

Формы записи условного оператора:

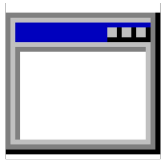
```
If <условие> Then <оператор>;
If <условие> Then <оператор_1> Else <оператор_2>;
```

- перед **Else** знак ; не ставится;
- операции сравнения: **>**, **<**, **=** , **<>**, **>=** , **<=** ;
- логические операции: **Not**, **Or**, **And**.

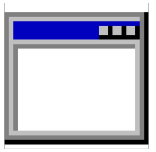
ПРИМЕР РАБОТЫ ПРОГРАММЫ



áèëåò.exe



ïðîâåðêà.exe



ïàëèíäðîì.exe

Оператор выбора Case

Оператор выбора используется для замены конструкций и вложенных условных операторов.

Синтаксис:

```
Case <порядковая_переменная>
of
   <значение_1> : <оператор_1> ;
   <значение_2> : <оператор_2> ;
   ……..
   <значение_N> : <оператор_N>;
Else <оператор_N+1>;
End;
```

Значение проверяется.

В операторе выбора можно использовать операторные скобки.

Не обязательная строка.

Оператор выбора Case

Для перечисления значений используется **запятая**, для выбора диапазона — **двоеточие**.

```
Case Размер одежды ученика of
  16..30 : Вы ученик начальных классов;
  31,32,33 : Вы учитесь в 5-6 классе;
  34..50 :  Вы старшеклассник ;
Else Вы явно не ученик;
End;
```

```
Case x of
  -128..-1: writeln('Отрицательные');
  0:        writeln('Ноль');
  1..127:      writeln('Положительные')
Else WriteLn('Выход из диапазона');
End;
```

ЗАДАНИЕ

1. **Выполните программу** «Калькулятор», которая при вводе символа с клавиатуры: «+», «−», «/», «*» выполняет соответствующие действия с двумя числами. Числа и символ операции вводятся с клавиатуры. Дайте анализ работы программы.

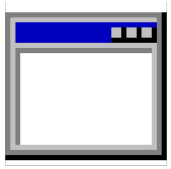
2. Обзор материала урока 14 «Ти ж мене пидманула, или Оператор выбора» электронного практикума.

Используемый материал:

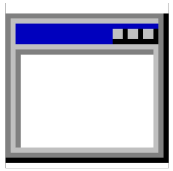
Оператор выбора:

```
Case <порядковая_переменная> of
     <значение_1> : <оператор_1> ;
     ……
     <значение_N> : <оператор_N> ;
Else <оператор_N+1> ;
End;
```

ПРИМЕР РАБОТЫ ПРОГРАММЫ



ãàäàíèå.exe



Proverka.exe

Циклические операторы

Цикл предусматривает многократное выполнение некоторых операторов, входящих в тело цикла.

В языке Pascal имеются три оператора цикла:

- `For` (цикл на заданное число повторений);
- `While` (цикл **ПОКА** — с предусловием);
- `Repeat` (цикл **ДО** — с постусловием).

Если **число повторений известно**, то лучше воспользоваться оператором цикла **с параметром**.

Цикл с параметром For

Цикл на заданное число повторений с *возрастающим* или *убывающим* значением параметра.

```
For {парам} := {нач_зн} To
               {кон_зн} Do
  {оператор} ;
```

Тело цикла

Замечания:

- параметр — порядковый тип;
- в цикле можно использовать операторные скобки;
- в теле цикла нельзя менять параметр цикла;
- параметр цикла увеличивается на единицу;
- начальное значение больше конечного, иначе тело цикла игнорируется;
- для уменьшения параметра, `to` заменяется на `downto`.

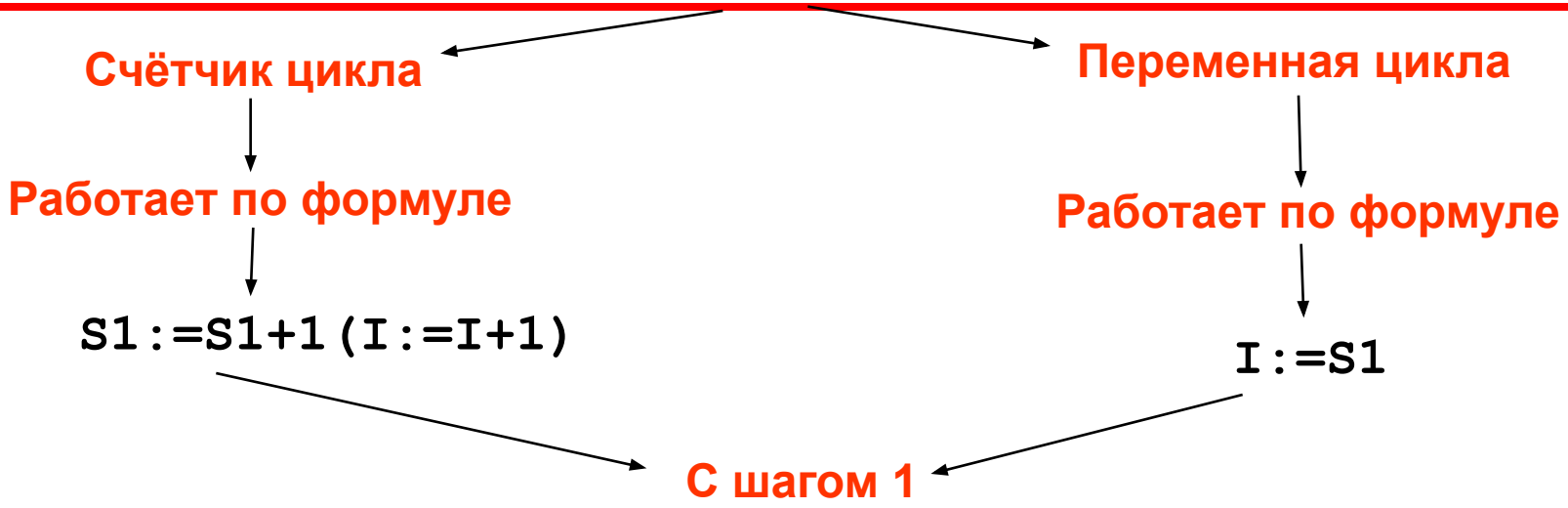
```
Program My_program;

Uses CRT;

Var f:Integer;

Begin
 ClrScr;

 For f := 1 to 4 do begin
   Write (f,'   ');
   WriteLn (f*f)
 End;

 ReadKey;
End.
```

```
1   1
2   4
3   9
4   16
```

Цикл с параметром For

Параметр цикла имеет 2 функции.

```
FOR <пар_цик>:=S1 to S2 do
```



Для изменения шага в цикле :

- вводят дополнительную переменную, отвечающую за изменение шага в алгоритме;
- изменяют значение переменной цикла, при этом учитывая:
  - целый тип;
  - конечное значение.

ЗАДАНИЕ

1. **Выполните задание** урока 11 «И получилась звёздная дорога, или Цикл с параметром» электронного практикума.

2. **Проверьте себя:** «Случайное число из промежутка» из урока 11 электронного практикума.

- Команда `random(n)` выдаёт случайное целое число из промежутка [0;n−1].
- Функция `random(16)` возвращает случайное число из промежутка длиной в 16 значений.
- В промежутке [9;15] содержится 15–(9–1)=7 значений; значит, нам нужна функция `random(7)` — она вернёт значение из промежутка [0;6].
- Чтобы «сдвинуть» этот промежуток до требуемого, достаточно к значению функции прибавить 9 — получится `random(7)+9`.

Вложенная структура циклического оператора For

1.Вывод таблицы умножения в столбец.

Внешний цикл J
Начинает работу.
Выполняется 10 раз.

Внутренний цикл I
Выполняет 10 проходов
за 1 проход внешнего цикла.
Выполняется 100 раз.

```
{таблица умножения}
uses crt;
var i,j:integer;
begin
clrscr;
for j:=1 to 10  do
 begin
  writeln;
    for i:=1 to 10 do
    write(j,' * ',i, ' = ',
 end;
readkey;
end.
```

```
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
7 * 10 = 70

8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72
8 * 10 = 80
```

Все операторы внутреннего цикла должны располагаться в теле внешнего.

Передача управления происходит от внутреннего цикла к внешнему!!!

ЗАДАНИЕ

1. **Выполните программу** вывода на экран в три столбца список чисел от 1 до N, их квадратов и кубов. Число N вводится с клавиатуры. Например, для N = 5 на экране должно быть:

1	1	1
2	4	8
3	9	27
4	16	64
5	25	125

Для проверки корректности работы программы при различных входных данных проводят её тестирование, которое заключается в подборе самых разнообразных входных данных, чтобы получить все возможные (и невозможные) варианты работы программы и «выловить» неучтённые ошибки.

2. **Выполните программу** вывода строчных букв латинского алфавита в прямом и обратном порядке.

Используемый материал:

Оператор цикла `For`:

```
For <парам> := <нач_зн> To <кон_зн> Do <оператор>;
```

- параметр – целый тип (обычно, `Integer`);
- в цикле можно использовать операторные скобки;
- параметр цикла увеличивается на единицу.

Цикл While («ПОКА»)

Цикл **While** сначала проверяет **условие**, и только если оно истинно, выполняет тело цикла.

```
While {условие} do
  {оператор};
```

- В теле кода, написанном ниже, цикл не выполнится ни разу:

```
x:=1;
While x>1 do
  x:=x-1;
```

- Можно получить бесконечный цикл. Например:

```
x:=1
While x>0 do
x:=x+1;
```

Цикл While («ПОКА»)

Программа вывода на экран суммы чисел от *a* до *b*.

Цикл работает, пока изменяющаяся переменная *f* не станет больше значения *b*.

```
Program My_program;

Uses CRT;

Var a,b,s,f:Integer;

Begin
 ClrScr;

 Write ('Введите начальное число: ');
 ReadLn (a);
 Write ('Введите конечное число: ');
 ReadLn (b);

 S:=0;

 F := a;
 While f<=b do Begin
   S := S + f;
   F := F + 1;
 End;

 Write ('Сумма чисел от ', a, ' до ', b, ' равна ', S);

 ReadKey;
End.
```

Можно ли обойтись без переменной F?

Попробуй изменить алгоритм.

```
Output
Введите начальное число: 2
Введите конечное число: 10
Сумма чисел от 2 до 10 равна 54
```

ЗАДАНИЕ

1. Выполните программу, которая определяет максимальное из введённых чисел с клавиатуры (ввод чисел заканчивается числом 0). Ниже представлен рекомендуемый вид экрана:

```
Введите числа. Для завершения ввода
введите 0.
89
15
0
Максимальное число 89.
```

Используемый материал:

Оператор цикла **While**:

While *<условие>* **do** *<оператор>;*

Цикл **While** сначала проверяет условие, и только если оно истинно, выполняет основное тело цикла.

Цикл Repeat («ДО ТЕХ ПОР»)

Цикл **Repeat** сначала выполняет тело цикла, а лишь затем проверяет условие.

```
Repeat
   {тело_цикла}
Until {условие};
```

Нет необходимости в цикле **Repeat** использовать составной оператор, т. к. данная конструкция предусматривает выполне-ние не одного, а нескольких операторов, заключённых между словами **Repeat** и **Until**.

Цикл Repeat («ДО ТЕХ ПОР»)

Пример программы вывода на экран суммы чисел от *a* до *b*.

```
Program My_program;

Uses CRT;

Var a,b,s,f:Integer;

Begin
 ClrScr;

 Write ('Введите начальное число: ');
 ReadLn (a);
 Write ('Введите конечное число: ');
 ReadLn (b);

 S:=0;

 F := a;
 Repeat
   S := S + f;
   F := F + 1;
 Until f>b;

 Write ('Сумма чисел от ', a, ' до ', b, ' равна ', S);

 ReadKey;
End.
```

```
Output
Введите начальное число: 2
Введите конечное число: 10
Сумма чисел от 2 до 10 равна 54
```

Цикл работает, пока изменяющаяся переменная *f* не станет больше значения *b*

ЗАДАНИЕ

1. Выполните программу «Калькулятор» (слайд 50). Применив цикл **Repeat**, доработайте программу самого калькулятора, где должен производиться запрос на продолжение работы или выхода из программы.

Используемый материал:

Оператор цикла **Repeat**:

```
Repeat <оператор> Until <условие>;
```

Цикл **Repeat** сначала выполняет основное тело цикла, а затем проверяет условие, и только если оно истинно, завершает свою работу.

Анализ работы трёх циклических операторов

For

```
For f:=a to b Do
   S := S + f;
```

While

```
F := a;
While f<=b do Begin
   S := S + f;
   F := F + 1;
End;
```

Repeat

```
F := a;
Repeat
   S := S + f;
   F := F + 1;
Until f>b;
```

- Выбор модели цикла, зависит лишь от удобства его использования в конкретной ситуации.
- Мы практически всегда можем вместо одного вида цикла воспользоваться другим.

- Инициализируем начальное значение.
- Наращиваем «счётчик цикла».

Зацикливание и прерывание циклов

For

```
{бесконечный цикл}
x:=0;
for i:=1 to 10 do
 begin
  x:=x+1;
  i:=1;
 end;
```

```
{прерывание цикла}
x:=0; i:=1;
while i<=10 do
 i:=i+10;
```

While

```
{бесконечный цикл}
x:=0; i:=1;
while i<=10 do
 x:=x+1;
```

```
{прерывание цикла}
x:=0;
for i:=1 to 10 do
 begin
  x:=x+1;
  i:=10;
 end;
```

Repeat

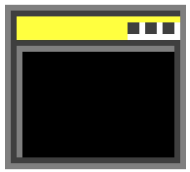
```
{бесконечный цикл}
x:=0; i:=1;
repeat
x:=x+1;
until i>=10
```

```
{прерывание цикла}
x:=0; i:=1;
repeat
i:=i+10;
until i>=10
```

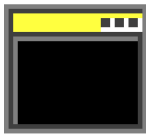
Для гибкого управления циклическими операторами используются процедуры:

- **Break** — выход из цикла;
- **Continue** — завершение очередного прохода цикла.

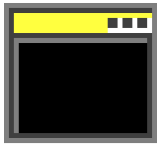
ПРИМЕР РАБОТЫ ПРОГРАММЫ



время.ex
e



пароль.ex
e



система_счисления.ex
e

Символьный тип данных (Char)

Символьный тип данных — для хранения одного символа

```
USES CRT;
VAR c:char;
BEGIN
clrscr;
      for c:='A' to 'z' do
        Write(c,' ');
        READKEY;
END.
```

```
A B C D E F G H I J K L M N O P Q R
i j k l m n o p q r s t u v w x y z
```

Один из 256 символов. Таблицы ASCII-кодов.

Значения в апострофах.

Буквы расположены подряд по алфавиту (**for**).

```
uses crt;
var  n: integer; c: char;
Begin
  clrscr;
  Repeat
    c:=readKey;
    n:=ord(c);
    writeln(n);
  Until n=27;
End.
```

```
80
50
13
27
```

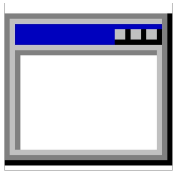
n:=#97;

1. Программа вывода на экран малых и больших букв латинского алфавита.
2. Программа определения числового значения ASCII-кода нажатой клавиши.

Операции с символьными переменными

Ord(x) — возвращает порядковый номер.
Chr(x) — преобразует целое число (тип `Byte`) в символ ASCII-кода.
Pread(x) — возвращает предыдущее значение.
Succ(x) — возвращает последующее значение.

Проверьте себя: Лабораторная работа



Lsim.exe

Символьный тип данных(Char)

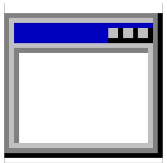
1. Выведите алфавит в столбец. Организуйте запрос на количество колонок (ширина поля алфавита).

```
uses crt;
VAR c:char;
    n,i:integer;
BEGIN
clrscr;
     Write('Ширина: '); ReadLn(n);
     c:='A';
     Repeat
           for i:=1 to n do
           begin
                if c in ['A'..'Z']
                   then Write(c,' ');
                inc(c);
           end;
           WriteLn;
     Until c>'Z';
     ReadLn;
END.
```

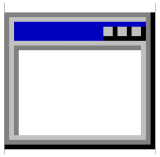
```
Ширина: 5
A B C D E
F G H I J
```

1. Цикл **For** определяет ширину поля.
2. Для вывода последовательности алфавита — используем диапазон.
3. <C> — начало диапазона.
4. Сравниваем <C> с элементом диапазона (**IN**) и выводим.
5. В <C> загружаем следующий элемент диапазона (**INC(C)**).
6. Сравниваем , выводим и т.д. до 7.
7. Пункт 5,6,7— в цикле **Repeat**.

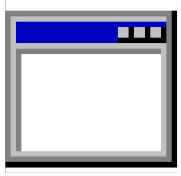
ПРИМЕР РАБОТЫ ПРОГРАММЫ



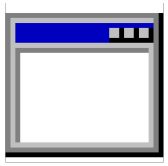
Ascii~1.exe



Matr.exe



Muzik~1.exe



Tar_bega.exe

ЗАДАНИЕ

1. **Выполните программу**, способную управлять движением объекта на экране (движение паучка).

2. **Обзор материала** урока 25 «Какой ты за собой оставишь след, или Типизированные константы» электронного практикума.

Используемый материал:

Символьный тип называется **Char**:

- символы заключаются в апострофы;
- буквы расположены согласно алфавиту в таблице ASCII-кодов.

Алгоритм моделирования движения

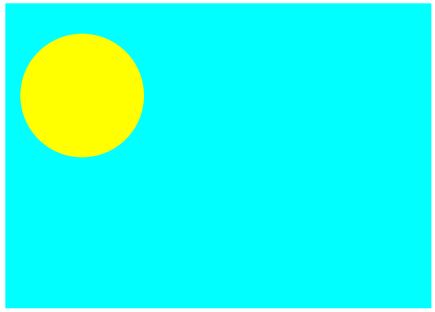
1. Устанавливаются начальные значения координат объекта.
2. В цикле объект стирается.
3. По формулам изменяются его координаты.
4. Объект выводится на экран (уже в новом месте).

Алгоритм моделирования движения объекта, изменяющего свою форму

1. Выводится форма.
2. Создаётся временная задержка.
3. Стирается форма.
4. Изменяются координаты объекта;
5. Алгоритм повторяется сначала.

Анимация в Pascal

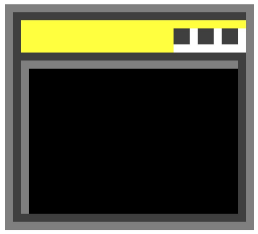
1. **Выполните программу** по анимации объекта. Движение солнца по небу.



2. **Выполните** задание урока 21 «Про маленькую гордую гусеницу, или Покадровая анимация» электронного практикума.

Контрольный обзор по разделу

Раздел 2. Основные алгоритмические конструкции		7 часов
Тема 4	Оператор ветвления	2 часа
Тема 5	Оператор повтора	2 часа
Тема 6	Символьный тип данных	2 часа
Тема 7	Графика. Анимация	1 час



ТЕСТ
Раздел 2

Массивы. Типы пользователя

Массив — это фиксированное количество значений одного типа.

Массив объявляется в разделе `Var`:

```
{Имя} : Array [ {нач_зн} .. {кон_зн} ] of {тип} ;
```

Структура одномерного массива:

A =

1	2	3	4
9	7	0	0

Индекс

Значение

Переменная массива (имя массива)

Доступ к массиву осуществляется через индекс:

<Имя массива> **[** ***<Индекс>*** **]**

A[i] := значение; → **A[2] := 7;**

Структура двумерного массива:

11	12	13	14
21	22	23	24
31	32	33	34
41	42	43	44

Индекс строки `i`

Индекс столбца `j`

Одномерные массивы

Примеры объявления массивов:

```
Var A : Array [1..4] of String;
    B : Array [0..662] of Real;
    C : Array [1..10] of Integer;
```

Примеры заполнения массивов значениями:

```
A[1]:='Вася'; A[2]:='Петя';
A[3]:='Маша'; A[4]:='Олеся';
Write (A[3]);

For f:=1 to 10 do C[f] := f*2;
For f:=1 to 10 do WriteLn ( C[f] );
```

Многомерные массивы


```
{Имя} : Array [ {нач_зн} .. {кон_зн} ,
                {нач_зн} .. {кон_зн}
,
                {и т.д.}
] {тип} ;
```

Список интервалов для каждой размерности массива.

Пример объявления двухмерного массива (матрицы, таблицы) на 4 строки и 6 столбцов:

```
Var A : Array [1..4,1..6] of Integer;
```

Пример заполнения массива:

```
For i:=1 to 4 do
  For j:=1 to 6 do
    A[i,j] := i+j;
```

$$A_{i,j} = \begin{bmatrix} 2 & 3 & 4 & 5 & 6 & 7 \\ 3 & 4 & 5 & 6 & 7 & 8 \\ 4 & 5 & 6 & 7 & 8 & 9 \\ 5 & 6 & 7 & 8 & 9 & 10 \end{bmatrix}$$

Организация ввода вывода данных в массивах

Ввод и вывод в одномерных массивах:

Ввод с клавиатуры

```
For i:=1 To N Do
Read(A[i])
```

Ввод случайным образом

```
For i:=1 To n Do
A[i]:=Random(m);
```

Ввод по формуле

```
For i:=1 To n Do
A[i]:=Sin(i);
```

Вывод на экран:

```
For i:=1 To N Do
Write(A[i]:3);
```

Ввод и вывод в многомерных массивах:

Ввод с клавиатуры

```
For i:=1 To n Do
For j:=1 To n Do
Read(A[i,j]);
```

Вывод на экран

```
For i:=1 To n Do
Begin
   For j:=I To n Do
   Write (A[i,j]);
   WriteLn;
End;
```

Стандартные алгоритмы над элементами массива

Накопление суммы

```
s:=0;
For i:=1 To n Do
 s:=s+a[i];
```

Накопление произведения

```
p:=1;
For i:=1 To n Do
 p:=p*a[i];
```

Перестановка элементов

```
n:=a[i];
a[i]:=a[i+1];
a[i+1]:=n;
```

Нахождение min (max) элемента и его порядковый номер

```
min:=a[i];
For i:=2 To n Do
 if  min>a[i] Then Begin min:=a[i]; minn:=i; End;
```

Удаление элемента

```
x-номер  удаляемого
элемента
For i:=1 To n Do
 If (i<>x) Then
  Writeln(a[i]:3);
```

или

```
FOR I:=x To n-1 Do
 a[i]:=a[i+1];
```

Учитывай ранг
Сдвиги меняют ранг.

Сортировка элементов

```
For i:=1 To n-1 Do
 If a[i+1]<a[i] Then
  Begin
   stek:=a[i];
   a[i]:=a[i+1];
   a[i+1]:=stek;
   i:=0;
  End;
```

Возврат на начало массива

Квадратные матрицы

Количество строк равно количеству столбцов

a11	a12	a13	a14
a21	a22	a23	a24
a31	a32	a33	a34
a41	a42	a43	a44

Побочная диагональ.
Сумма индексов элементов на 1 больше размерности строки/столбца.

Главная диагональ `i=j`

a11	a12	a13	a14
a21	a22	a23	a24
a31	a32	a33	a34
a41	a42	a43	a44

Для элементов над главной диагональю `i<j`

Для элементов под главной диагональю `i>j`

a11	a12	a13	a14
a21	a22	a23	a24
a31	a32	a33	a34
a41	a42	a43	a44

Исходная

a11	a21	a31	a41
A12	a22	a32	A42
A13	A23	a33	a43
a14	a24	a34	a44

Преобразованная

```
For i:=1 to 5 do
For j:=i+1 to 5 do
 Begin
  N:=a [i,j];
  A [i,j:=a[j,i];
  A [j,i:=n;
 End;
```

Генератор случайных чисел (ГСЧ)

Для генерации в программе случайных чисел используют генератор случайных чисел (ГСЧ) — функция **random**

Randomize — инициализация ГСЧ. Объявляется только в самом начале программы. Различают два вида **random**:

- **с целым параметром** **random(n)** — возвращает целое случайное число из промежутка **[0;n-1]**;
- **без параметра** **random** — возвращает вещественное случайное число из промежутка **[0;1]**.

Для получения целого случайного числа из произвольного промежутка **[a;b]** *используется формула:* **a+random(b-a+1)**.

```
Program My_program;

Uses CRT;

Var n,f:Integer;
    A : Array [1..7] of Integer;

Begin
 ClrScr;
 Randomize;

 Write ('Случайные числа от 0 до ');
 Read (n);

 For f:=1 to 7 do begin
   A[f] := random (n+1);
   Write (A[f],' ')
 end;

 ReadKey;
End.
```

```
Output
Случайные числа от 0 до 10
7 2 7 3 0 7 6 _
```

ЗАДАНИЕ

1. **Выполните программу**, которая заполняет двухмерный массив случайными числами от −10 до 20 и сортирует значения массива по возрастанию.

2. Выполните упражнение урока 25 «Какой ты за собой оставишь след, или Типизированные константы» электронного практикума.

Используемый материал:

Объявления массива:

```
<Имя> : Array [<нач_зн> … <кон_зн>] of <тип>;
```

Доступ к массиву:

```
<Имя массива> [<Индекс>] ;
```

Строковый тип данных (String)

Строки — упорядоченный набор символов.

```
Program My_Program;

Uses CRT;

Var S,Z:String;
    x:Integer;

Begin

 ClrScr;

 X := 10;
 S := 'Вася';
 Z := 'Меня зовут ' + S;

 WriteLn( Z );
 WriteLn( 'Мне ', X, ' лет' );

 ReadKey;

End.
```

```
Меня зовут Вася
Мне 10 лет
```

```
S := X;
X := S;
```

Тип	Диапазон
String	255 символов

Строки не совместимы с целыми и вещественными типами.

Строки заключены в апострофы.

{ Основные операторы для строк }

- **+** *{ конкатенация }*
- **Length (S);** *{ длина строки }*

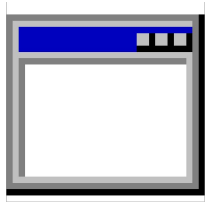
Например:

```
X := 'Вася';
Write( 'В вашем имени',
       Length(X),
       'букв.');
```

Операции со строковыми переменными

Delete — удаление символа из строки.
Insert — вставка строки в строку.
Copy — выделение подстроки.
Concant — выполнение сцепления строк.
Pos — обнаружение первого появления в строке подстроки.
UpCasae — преобразование строчной букву в прописную.
Str — преобразование числового значения величины в строку.
Val — преобразование значения строки в величину целочисленного или вещественного типа.

Проверьте себя: Лабораторная работа



Labstr.exe

ЗАДАНИЕ

1. **Выполните программу** вывода слова наоборот и подсчитайте количество слов в строке.

2. **Создайте самостоятельно программу** с использованием процедуры **`Str`** .

3. **Создайте самостоятельно программу** с использованием процедуры **`Val`**.

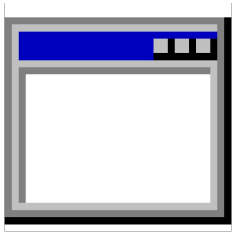
4. **Обзор программ** урока 27 « Шоу бегущих строк, или Этюды об одном типе данных» электронного практикума.

Используемый материал:

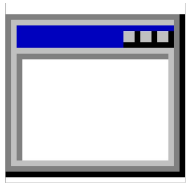
Строковый тип называется **`String`**

- строковая переменная может содержать не более 255 символов;
- если заранее известно, что длина строковой переменной не будет превышать некоторого значения **`n`**, то её объявляют как **`string[n]`**.

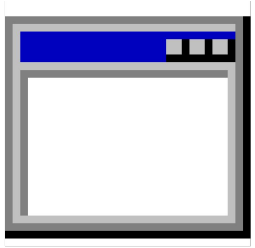
ПРИМЕР РАБОТЫ ПРОГРАММЫ



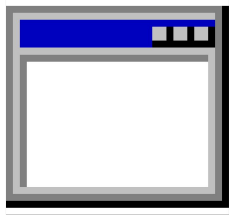
Begstr.exe



Kolslov.exe



Val.exe



Str.exe

Работа с файлами

Компилятор Turbo Pascal поддерживает три типа файлов:

Текстовый — определяются типом `text`, например: `<Имя> : Text;`	`Var f:text;`

Типизированный — для хранения однородной информации, например числовых данных одного типа.
Задаются предложением:
`file of <тип>,`
например:
`<Имя>= File Of <тип>;`
	`Var`
	`F:file of integer;`
	`или`
	`Type`
	`Fl=file of integer;`
	`Var F:fl;`

Нетипизированный — для хранения разнородной информации, тип которой может меняться или неизвестен.
Определяются типом `file`, например:
`<Имя>= File;`
	`Var F2:file;`
	`или`
	`Type`
	`Fl=file;`
	`Var F2:fl;`

Работа с файлами

Последовательность действий при работе с файлами:

1.Объявление файловой переменной (ФП), например, текстовой:

Var (*<ФП>*:*<тип файла>*);

```
Var f1,f2,f3,f4:text;
```

2. Ассоциация ФП с файлом:

Assign (*<ФП>*, *<имя файла>*);

```
Assign (f,'C:\my\Data.ghm');
```

3. Открытие файла для чтения/записи:

Reset(Rewrite)(*<ФП>*); — открывает файл для чтения (записи).

4. Операции с файлом:

Read (*<ФП>*, *<П1>* , *<П2>* , …); — считывает в переменные *<П1>*, *<П2>* и т. д. по одному элементу с позиции указателя;

Write (*<ФП>* , *<П1>* , *<П2>* , …); — записывает в файл значения переменных *<П1>*, *<П2>* и т. д. по одному элементу с позиции указателя;

EoLn(EoF) (*<ФП>*); — возвращает **True**, если достигнут конец строки (Файла);

Append (*<ФП>*); — открывает существующий текстовый файл для добавления данных в конец файла.

5. Закрытие файла: **Close** (*<ФП>*).

ЗАДАНИЕ

1. Создайте программу, которая считывала бы информацию с трёх исходных файлов, обрабатывала её и выдавала записи в результирующий файл по следующему принципу:

<имя в им.пад> <чувство/действие> <имя в вин. пад>;

файл	Содержимое файла
Первый исходный файл	*Имена в именительном падеже*
Второй исходный файл	*Имена в именительном падеже*
Третий исходный файл	*Список выражения чувств или какое-то действие, которое может произойти между участниками*
Результат	**Ольга любит Сергея** **Олег хочет видеть Романа** **Катя уважает Настю** *И т.д.*

Используемый материал:

Для организации работы текстовыми с файлами используют процедуры:

Assign, Rewrite, Reset, Write, Read, Close.

Множества

Множество — неупорядоченная совокупность неповторяющихся элементов одного типа, имеющих общее имя.

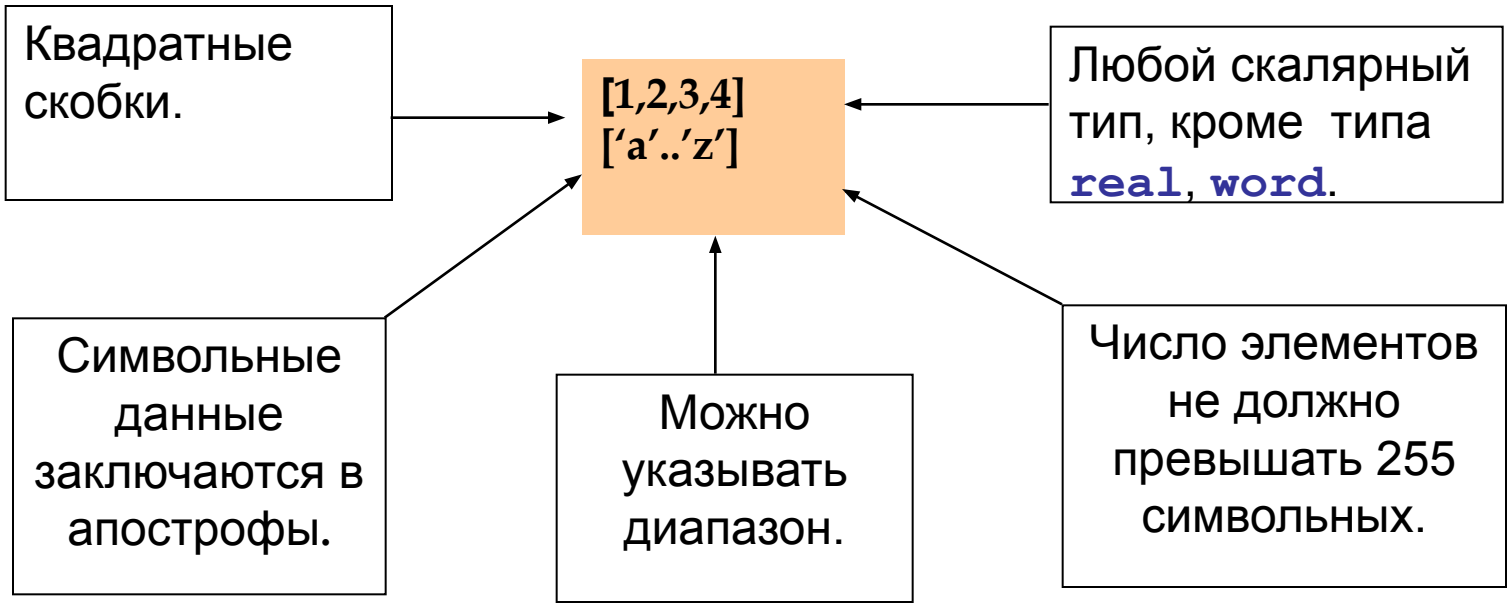
Синтаксис:

```
<имя типа > = SET OF <базовый тип>
```

Например:

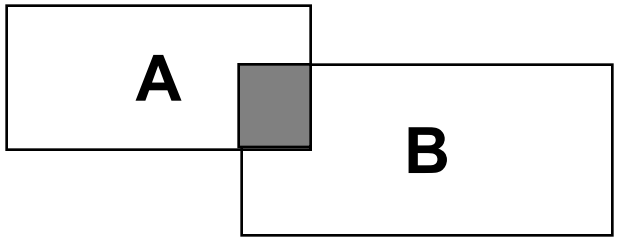
```
VAR
   M:set of char; st:string;
BEGIN
   M:=['a'..'z','0'..'9'];
```

Правила записи элементов множества

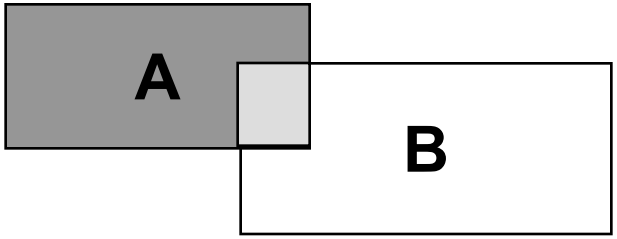


Операции над множествами

Объединение множеств (A+B)



Пересечение множеств (A*B)



Разность множеств (A−B)

ЗАДАНИЕ

1. **Выполните программу:** Из множества целых чисел 1...20 выделите множество чисел, делящихся на 6, на 2 и на 3 без остатка. Выведите содержимое этих множеств на экран.
2. **Выполните программу:** Введите строку символов, состоящую из латинских букв, цифр, пробелов. Осуществите проверку правильности введённых символов.

Используемый материал:

```
TYPE   <имя типа > =  SET OF
            <элемент 1,…,элемент n>
VAR    <идентификатор >:<имя типа>;
```

Записи

Запись — это набор элементов разнородного типа. Элементы (поля) определяются именем. Доступ к конкретному полю происходит через обращение имени записи и имени поля.

Синтаксис записи

```
TYPE
< имя_записи> = record
                    <имя_поля>: <тип_данных>;
                                .....;
                <имя_поля>: <тип_данных>;

                    end;
VAR
  <имя_переменной>:< имя_записи>;
```

Доступ

<имя_записи>. <имя_поля>

Записи

Запись — это набор элементов (полей) разнородного типа.

Пример: **persona**

name	sex	date			sb
		day	moth	year	
Иванов	м	23	апрель	1997	4

II Тип записи
Тип — *tovar*

I Тип записи
Тип — *date*

```
type
  date= record
      day:string[20];   {день}
      month:real;       {месяц}
      year:byte;        {год}
  end;
```

```
type
  persona= record
     name:string[20];{фамилия}
     sex:boolean;        {пол}
     bd: date;      {дата рожд.}
     sb:real;       {средний балл}
  end;
var
   student: persona;
```

`student` — переменная записи типа — `persona`;

```
student.name:='Иванов';
student.sex:='м';
student.bd.day:=23;
student.bd.month:='апрель';
```

Обращение к полю через комплексное имя

Записи

Доступ к конкретному полю происходит через обращение имени записи и имени поля.

Существуют два вида обращения:

Имя записи. Имя поля

With
(оператор присоединения)

```
student.name:='Иванов';
student.sex:='м';
student.bd.day:=23;
student.bd.month:='апрель';
```

```
With student do
  begin
   name:='Иванов';
   sex:='м';
   bd.day:=23;
   bd.month:='апрель';
  end;
```

ЗАДАНИЕ

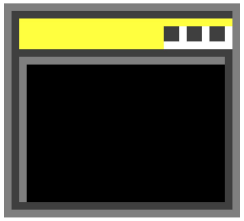
1. Выполните задание урока 28 «Живут студенты весело, или Записи» электронного практикума.

Используемый материал:

```
 Type
  < имя_записи> = record
                <имя_поля>: <тип_данных>
              . . . .;
                <имя_поля>:
<тип_данных>;

  end;
Var
    <имя_переменной>:< имя_записи>;
```

Контрольный обзор по разделу

Раздел 3. Структурированные типы		8 часов
Тема 8	Массивы	2 часа
Тема 9	Строки	1 час
Тема 10	Множества и записи	2 часа
Тема 11	Работа с файлами	2 часа



ТЕСТ

Раздел 3

Разработка проекта

Для разработки проекта выберите один из вариантов:

Задание 1 к уроку 29 «Заметка на память, или Типизированные файлы» электронного практикума.

Урок 30 «Графический редактор, или Работа с текстовыми файлами» электронного практикума.

Откройте любую программу с рисунком и организуйте чтение картинки из файла.