# Язык программирования Pascal

ПРЕЗЕНТАЦИЮ ПОДГОТОВИЛА ШИКУНЕЦ ВЕРОНИКА 9Б

# История создания

- Язык Паскаль был создан Никлаусом Виртом в 1968—1969 годах после его участия в работе комитета разработки стандарта языка Алгол-68. Язык назван в честь французского математика, физика, литератора и философа Блеза Паскаля, который создал первую в мире механическую машину, складывающую два числа. Первая публикация Вирта о языке датирована 1970 годом; представляя язык, автор в качестве цели его создания указывал построение небольшого и эффективного языка, способствующего хорошему стилю программирования, использующему структурное программирование и структурированные данные.
- Последующая работа Вирта была направлена на создание на основе Паскаля языка системного программирования, с сохранением возможности вести на его базе систематический, целостный курс обучения профессиональному программированию.

# Никлаус Вирт

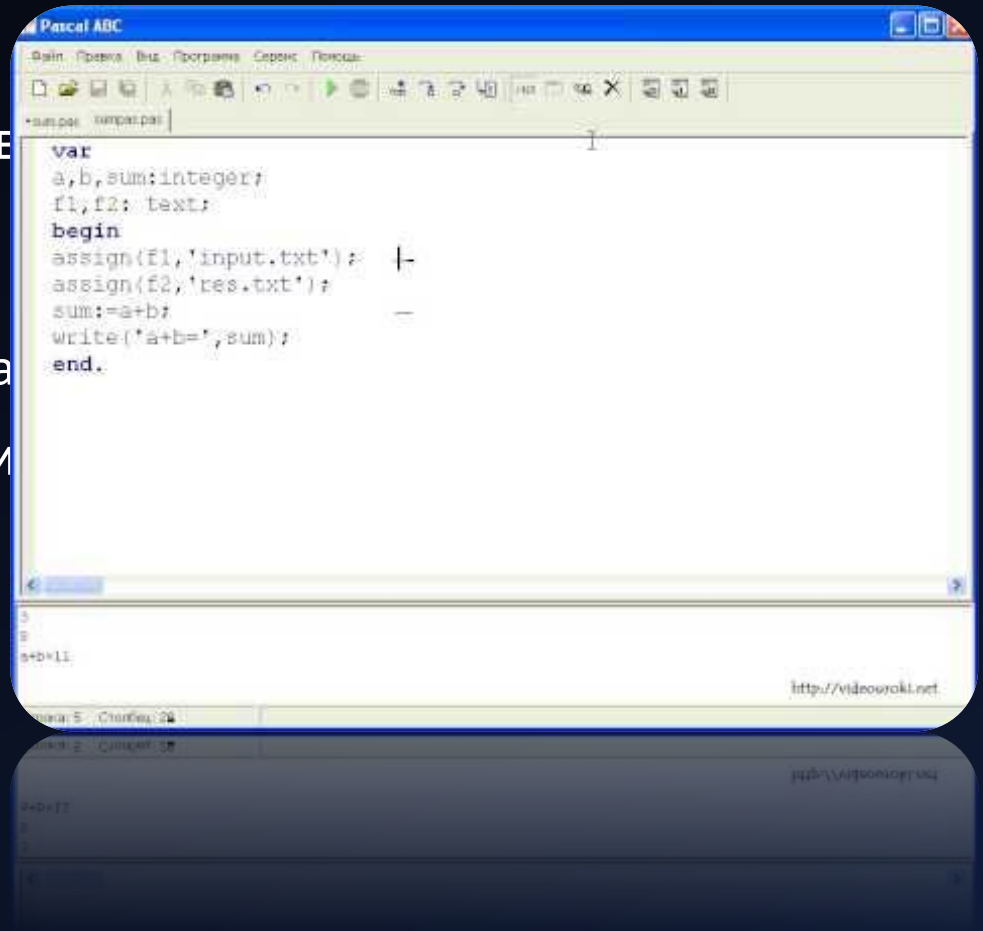
# Реализации и диалекты

- В 1978 году в Калифорнийском университете в Сан-Диего была разработана система UCSD p-System, включавшая порт компилятора Вирта с языка Паскаль в переносимый p-код, редактор исходных кодов, файловую систему.(**UCSD Pascal)**
- В 1986 году фирма Apple разработала объектное расширение языка Паскаль, получив в результате Object Pascal. Он был разработан группой Ларри Теслера, который консультировался с Никлаусом Виртом.(**Object Pascal**)


- В 1983 году появилась первая версия интегрированной среды разработки Turbo Pascal фирмы Borland, основывавшаяся на одноимённой реализации Паскаля.
- В 1989 году объектное расширение языка было добавлено в Turbo Pascal версии 5.5.
- Последняя версия (7.0) была переименована в Borland Pascal.
- Дальнейшее развитие реализации Паскаля от Borland породило вариант Object Pascal от Borland, впоследствии, в ходе развития среды программирования Delphi, получивший одноимённое название.

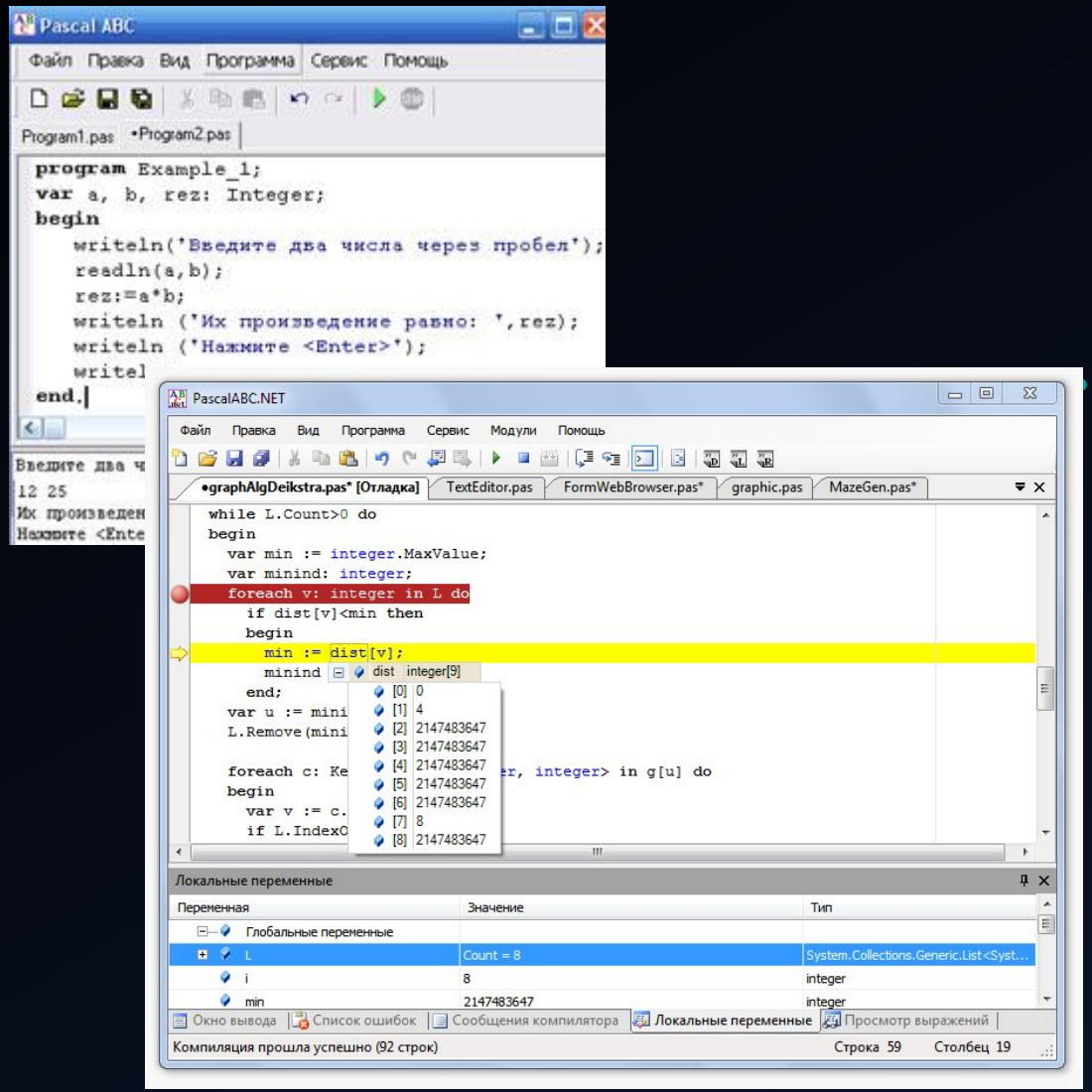
# Современные версии Object Pascal

- Важным шагом в развитии языка является появление свободных реализаций языка Паскаль Free Pascal и GNU Pascal, которые не только вобрали в себя черты множества других диалектов языка, но и обеспечили чрезвычайно широкую переносимость написанных на нём программ (например GNU Pascal поддерживает более 20 различных платформ, под более чем 10 различными операционными системами, Free Pasca обеспечивает специальные режимы совместимости с различными распространёнными диалектами языка, такими как Turbo Pascal (полная совместимость), Delphi и другими).




- в Южном федеральном университете разработан PascalABC.NET — язык программирования Паскаль, включающий большинство возможностей языка Delphi, а также ряд собственных расширений. Он основан на платформе Microsoft.NET и содержит практически все современные языковые средства: классы, перегрузку операций, интерфейсы, обработку исключений, обобщенные классы и подпрограммы, сборку мусора, лямбда-выражения.

# Особенности языка




- Особенностями языка являются строгая типизация и наличие средств структурного (процедурного) программирования. Паскаль был одним из первых таких языков. По мнению Вирта, язык должен способствовать дисциплинированному программированию, поэтому, наряду со строгой типизацией, в Паскале сведены к минимуму возможные синтаксические неоднозначности.

- Тем не менее, первоначально язык имел ряд ограничений: невозможность передачи функциям массивов переменной длины, отсутствие нормальных средств работы с динамической памятью, ограниченная библиотека ввода-вывода, отсутствие средств для подключения функций, написанных на других языках, отсутствие средств раздельной компиляции и т. п. Подробный разбор недостатков языка Паскаль того времени был выполнен Брайаном Керниганом в статье «Почему Паскаль не является моим любимым языком программирования». Некоторые недостатки Паскаля были исправлены в ISO-стандарте 1982 года, в частности, в языке появились открытые массивы, давшие возможность использовать одни и те же процедуры для обработки одномерных массивов различных размеров.

# Стандарты

- После начала использования Паскаля в 1970 году и появления реализаций, расходящихся не только в дополнениях, но и в синтаксисе, был поднят вопрос о стандартизации языка. Стандарт языка был разработан Никлаусом Виртом в 1974 году совместно с Кетлин Йенсен (Kathleen Jensen).[111] В дальнейшем, были приняты международный стандарт от ISO и американский от ANSI. На данный момент, выделяют три принципиально разных стандарта: Unextended Pascal (исходный), Extended Pascal (расширенный), Object-Oriented Extensions to Pascal (объектно-ориентированное расширение Паскаля). Одним из главных дополнительных свойств объектно-ориентированного расширения Extended Pascal стала модульность и средства, облегчающие раздельную компиляцию.
- Стандартизация языка была запаздывающей по отношению к реальному появлению в языке тех или иных возможностей. Коммерческие реализации расширяли стандартный Паскаль; так было сделано в UCSD Pascal, модификации Object Pascal фирмой Apple, Turbo Pascal от Borland (незначительно модифицированная версия Apple) и его ответвлений. Ни одна из распространённых коммерческих реализаций Паскаля не соответствует в точности ни одному из официальных стандартов языка.

# Hello,world!

- Приведённая выше программа не выполняет никаких действий и содержит пустой блок операторов.
- Пример программы, выводящей строку «Hello, world!»:

```
begin
  writeln('Hello, World!');
{ оператор вывода строки }
end.
```

# Синтаксис и языковые конструкции

- Паскаль, в его первоначальном виде, представляет собою чисто процедурный язык и включает в себя множество алголоподобных структур и конструкций с зарезервированными словами наподобие if, then, else, while, for, и т. д.
- В современных диалектах (Free Pascal) доступны такие операции, как перегрузка операторов и функций.
- Программы на Паскале начинаются с ключевого слова Program и следующего за ним имени программы с точкой с запятой (в некоторых диалектах является необязательным), за именем может в скобках следовать список внешних файловых дескрипторов («окружение») в качестве параметров; за ним следует тело программы, состоящее из секций описания констант (Const), типов (Type), переменных (Var), объявлений процедур (Procedure) и функций (Function) и следующего за ними блока операторов, являющегося точкой входа в программу. В языке Паскаль блок ограничивается ключевыми словами begin и end. Операторы разделяются точками с запятой, после тела помещается точка, служащая признаком конца программы.
- Регистр символов в Паскале не имеет значения.
- Таким образом, простейшая («пустая») программа на Паскале будет выглядеть следующим образом

```
program p;
begin
end.
```

# Типы данных

## Простые типы

1) *В стандартном и расширенном Паскале есть такие простые типы: числа с плавающей запятой (real), целые (integer), символьный (char), логический (boolean) и перечисления (конструктор нового типа, введённый в Pascal).*

2) *Turbo Pascal дополнил язык вариациями этих типов: например, shortint будет короче integer, а longint — длиннее.*

3) *Современные диалекты Pascal, такие, как FPC или Delphi, считают, что integer — это наиболее подходящий для данной машины целый, применяемый, например, для индексов массива, а shortint, longint и другие — целые определённой длины; это удобно при кроссплатформенном программировании. Аналогично и с дробными числами.*

4) *Ещё раз расширили типы при переходе на x64 — «просто целое» (integer) осталось 32-битным, но потребовался особый тип, который равен longint на x86 и int64 на x64.*

```
var { секция объявления переменных }
  r: Real;  { переменная вещественного типа }
  i: Integer;  { переменная целого типа }
  c: Char;  { переменная-символ }
  b: Boolean;  { логическая переменная }
  s: String; { переменная строки }
  t: Text; { переменная для объявления текстового файла }
  e: (apple, pear, banana, orange, lemon);  { переменная типа-перечисления }
```

- В Pascal над целыми типами (byte, shortint, word, integer, longint и их диапазоны) допустимы побитовые операции. Над битами двух целых операндов можно выполнять ранее рассмотренные логические операции: not, and, or, xor. Отличие между побитовыми и логическими операциями состоит в том, что побитовые (поразрядные) операции выполняются над отдельными битами операндов, а не над их значением в десятичном (обычно) представлении. Выделяется понятие порядковых типов данных (ordinal), к ним относятся целые типы (знаковые и беззнаковые), логический (boolean), символьный (char), перечислимые типы и типы-диапазоны.
- Для порядковых типов определены операции inc, dec, succ, pred, ord, операции сравнения (= > < => <= <>), их можно использовать в операторах case, for (как счётчик цикла), как границы массивов, для задания элементов множеств и типов-диапазонов.
- В Pascal, в отличие от Си-подобных языков, с типами boolean и char арифметические целочисленные операции не определены.

Диапазоны содержат подмножество значений других порядковых типов:

```
var
  x: 1..10;
  y: 'a'..'z';
  z: pear..orange;
```

отличие от многих распространённых языков, Pascal поддерживает специальный тип данных множество:

Множество — фундаментальное понятие в современной математике, которое может быть использовано во многих алгоритмах.

В паскале тип множество может содержать только однотипные элементы порядкового типа. Эта особенность широко используется и обычно быстрее эквивалентной конструкции в языке, не поддерживающем множества. К примеру, для большинства компиляторов Паскаля:

if i in [5..10] then { проверка на принадлежность элемента множеству }

обработается быстрее, чем

if (i>4) and (i<11) then { проверка логическими условиями }

```
var
  set1: set of 1..10;
  set2: set of 'a'..'z';
  set3: set of pear..orange;
```

# Составные типы

- c = File of a;  { определение файла }
- Файловые типы в Паскале делятся на типизированные, текстовые и файлы без типов.
- Как показано в вышеприведённом примере, типизированные файлы в Паскале — это последовательности однотипных элементов. Для каждого файла существует переменная-указатель на буфер, которая обозначается f^. Процедуры get (для чтения) и put (для записи) перемещают указатель к следующему элементу. Чтение реализовано так, что read(f, x) представляет собою то же, что и get(f); x:=f^. Соответственно, запись реализована так, что write(f, x) представляет собою то же, что и f^ := x; put(f).

Новые типы могут быть определены из существующих:

type { секция объявления типов }
  x = Integer;
  y = x;

Более того, из примитивных типов могут быть сконструированы составные:

```
type { секция объявления типов }
  a = Array [1..10] of Integer;  { определение массива }
  b = record  { определение записи }
        x: Integer;
        y: Char;
      end;
```

- Файловые типы в Паскале делятся на типизированные, текстовые и файлы без типов.
- Как показано в вышеприведённом примере, типизированные файлы в Паскале — это последовательности однотипных элементов. Для каждого файла существует переменная-указатель на буфер, которая обозначается f^. Процедуры get (для чтения) и put (для записи) перемещают указатель к следующему элементу. Чтение реализовано так, что read(f, x) представляет собою то же, что и get(f); x:=f^. Соответственно, запись реализована так, что write(f, x) представляет собою то же, что и f^ := x; put(f). Текстовые файлы text определены как расширение типа file of char и помимо стандартных операций над типизированными файлами (чтение, запись символа), позволяют осуществлять символьный ввод-вывод в файл всех типов данных аналогично консольному вводу-выводу.
- Файлы без типов объявляются как переменные типа file. С ними можно проводить операции побайтового нетипизированного ввода-вывода по несколько блоков байт указанной длины через буфер, для этого служат специальные процедуры blockread и blockwrite (расширение UCSD).

# Модули

- До появления связных модулей в их современном виде некоторые реализации Паскаля поддерживали модульность за счёт механизма включения заголовочных файлов, похожего на механизм #include в языке Си: с помощью специальной директивы, оформляемой в виде псевдокомментария, например, {$INCLUDE "файл"}, содержимое указанного файла прямо включалось в текст программы в исходном, текстовом виде. Таким образом можно было разделить программный код на множество фрагментов, для удобства редактирования, но перед компиляцией они автоматически объединялись в один файл программы, который в итоге и обрабатывался компилятором. Такая реализация модульности примитивна и имеет множество очевидных недостатков, поэтому она была быстро заменена.

- Современные реализации языка Паскаль (начиная с UCSD Pascal) поддерживают модули. Программные модули могут быть двух видов: модуль главной программы, который, как обычно, начинается с ключевого слова program и тело которого содержит код, запускаемый после загрузки программы в память, и вспомогательных модулей, содержащих типы, константы, переменные, процедуры и функции, предназначенные для использования в других модулях, в том числе в главном модуле.

# Структура

Тело модуля начинается находящимся на верхнем уровне вложенности ключевым словом BEGIN. Тело содержит программный код, который выполняется один раз при загрузке модуля. Тело может применяться для инициализации, присваивания начальных значений переменным модуля, выделения ресурсов для его работы и так далее. Тело модуля может отсутствовать. В ряде реализаций Паскаля, например, в Delphi, вместо тела модуля могут применяться две секции (также необязательные) — INITIALIZATION и FINALIZATION. Они располагаются в конце модуля, после соответствующего ключевого слова. Первая — секция инициализации, — содержит код, который должен быть выполнен при загрузке модуля, вторая — секция финализации, — код, который будет выполнен при выгрузке модуля. Секция финализации может выполнять действия, обратные инициализации — удалять объекты из памяти, закрывать файлы, освобождать выделенные ресурсы.

Модуль заканчивается ключевым словом END с точкой.

Общая структура подключаемого модуля на Паскале выглядит следующим образом:

```
unit UnitName1;
interface
  ...

implementation
  ...

begin {может отсутствовать - используется, если необходимо поместить операторы инициализации}
  ...
end.
```

Возможен также ещё один вариант:

```
unit UnitName2;
interface
  ...

implementation
  ...

initialization
  ...

finalization
  ....

end.
```

# Строки

- В современном Паскале для работы со строками используется встроенный тип string, поддерживающий операции конкатенации (+) и сравнения (> < = <> >= <=). Строки сравниваются в лексикографическом порядке. Например, строки считаются равными, если они имеют одинаковую длину и коды всех символов с одинаковыми индексами совпадают.
- Тип string [n] или просто string в диалектах языка 1970—1990-х годов определялся в виде массива символов array [0..n] of char (n по умолчанию принимало значение 80 в UCSD Pascal и 255 в Turbo/Borland Pascal), нулевой элемент массива при таком представлении служит для задания длины строки, соответственно строка могла иметь максимальный размер 255 символов. По умолчанию в Delphi и FreePascal в качестве String используется тип AnsiString, память под который выделяется и освобождается компилятором динамически, а максимальный размер строки в текущих реализациях составляет 2 гигабайта. Кроме того, в Delphi и Free Pascal в качестве string может использоваться тип UnicodeString, где применяется 16-битное представление символов в кодировке UCS-2, при этом средства преобразования из однобайтовых строк в многобайтовые и обратно в стандартной библиотеке FPC отсутствуют, но имеются в Delphi.

В Delphi 2009 и выше имеется конструкция для объявления AnsiString с определенной кодовой страницей:

```
type
  CyrillicString =
AnsiString(1251);
  CP866String =
AnsiString(20866);
```

# Указатели

- Здесь переменная pointer_to_b — указатель на тип данных b, являющийся записью. Типизированный указатель может быть определён (опережающее определение) перед объявлением типа, на который он ссылается. Это одно из исключений к правилу, которое гласит, что любой элемент (константа, тип, переменная, процедура, функция) должен быть объявлен перед тем, как используется. Введение этого исключения позволяет организовывать рекуррентные определения структур данных, в том числе такие, как линейные списки, стеки, очереди, деревья, включая указатель на запись в описание этой записи (см. также: нулевой указатель — nil).
- Для типизированного указателя определена операция разыменования (её синтаксис: указатель^).
- Чтобы создать новую запись и присвоить значение 10 и символ A полям x и y в ней, необходимы следующие операторы:

Паскаль поддерживает использование указателей (типизированные ^тип и нетипизированные pointer):

```
type
  a = ^b;
  b = record
        x: Integer;
        y: Char;
        z: a;
      end;
var
  pointer_to_b:a;
```

Для целей обращения к полям записей и объектов можно также использовать оператор with, как показано в примере:

```
new(pointer_to_b);

with pointer_to_b^ do
begin
  x := 10;
  y := 'A';
  z := nil
end;
...
dispose(pointer_to_b);
```

# Операторы управления

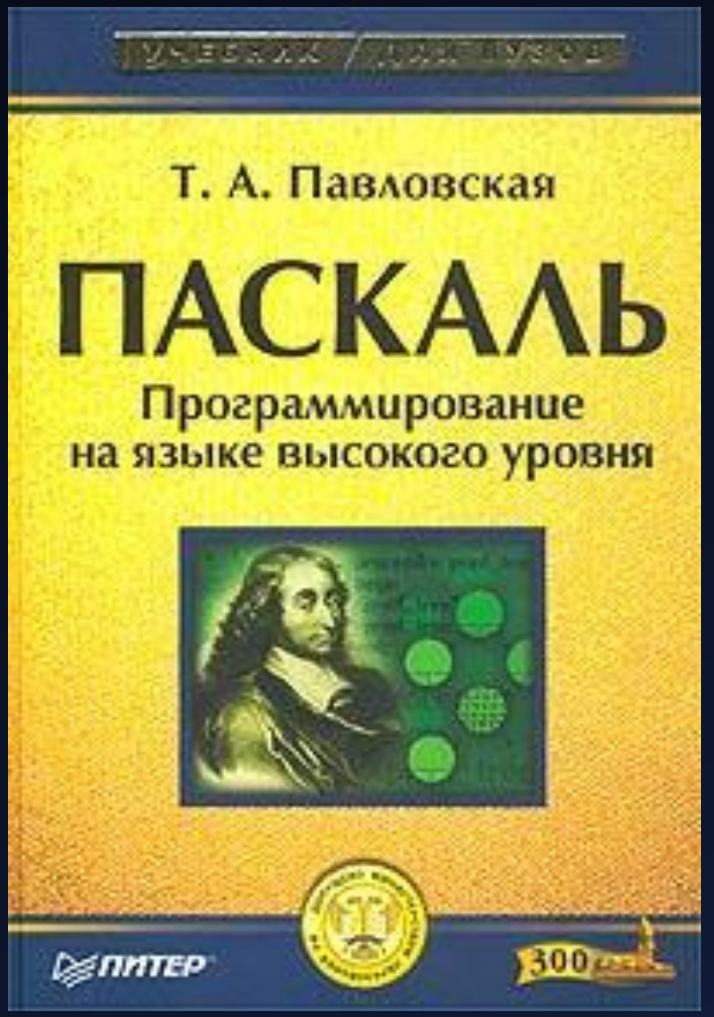
- Паскаль — язык структурного программирования, что означает, что программа состоит из выполняющихся последовательно отдельных стандартных операторов, в идеале — без использования команды GOTO.
- В операторах while, for, if, case в качестве выполняемого оператора может использоваться блок. Такая конструкция, представляющая собой обычный оператор или блок, называется сложным оператором.
- В Turbo Pascal для управления процессом компиляции существуют директивы, которые помещаются в комментарии и позволяют переключать режимы работы компилятора — например, включать и отключать проверку операций ввода-вывода, переполнения:

# Объектно-ориентированное программирование


- В Object Pascal имеется возможность разрабатывать программы с применением парадигмы объектно-ориентированного программирования. Классы задаются с помощью типа object, аналогичного record, который кроме полей данных может содержать заголовки процедур и методов. Имена описываемых методов следуют за именем класса через точку.
- Конструктор и деструктор задаются как обычные процедуры, но вместо идентификатора procedure задаются ключевые слова constructor и destructor. Соответственно, в отличие от C++-подобных языков они имеют имя, отличное от имени класса, деструкторов может быть несколько и они могут иметь параметры (на практике эта возможность используется редко, обычно класс имеет единственный деструктор Destroy, переопределяющий виртуальный деструктор класса-родителя).
- Поддерживаются единичное наследование, полиморфизм классов, механизм виртуальных методов (слово virtual после заголовка метода класса). Существуют и динамические методы (в TP описываются путём добавления целого числа после слова virtual и используются преимущественно для обработки сообщений; в Delphi и FreePascal для этих целей используется слово message, а для создания обычных динамических методов — слово dynamic), отличающиеся меньшим использованием памяти и меньшей скоростью вызова за счёт отсутствия дублирования динамических методов предков в VMT потомка (однако FreePascal не делает различий между виртуальными и динамическими методами).

# Пример программы для Pascal

- В диалекте Delphi классы могут также конструироваться с помощью слова class (причём взаимное наследование с object-классами не допускается) и введены интерфейсы (interface) — все методы абстрактные и не могут содержать полей данных.
- Все классы (созданные с помощью class) являются наследниками TObject, все интерфейсы происходят от IUnknown. Классы, созданные с помощью class, могут реализовывать несколько интерфейсов.
- В Delphi интерфейсы были введены для поддержки технологии COM фирмы Microsoft.
- Классы (Class) в отличие от обычных классов (Object) не нуждаются в явном выделении/освобождении памяти, память под них динамически выделяется конструктором с именем Create, вызываемым с именем класса, и освобождается при вызове деструктора с именем Destroy (могут иметь другие имена). Переменная такого класса в отличие от класса object хранит адрес экземпляра класса в памяти, значение nil используется для указания пустой ссылки, поэтому для освобождения объекта в TObject определен специальный метод free, проверяющий ссылку на nil и вызывающий виртуальный деструктор Destroy



# Конец

СПАСИБО ЗА ВНИМАНИЕ