Основы алгоритмизации

Наумов Д.А., доц. каф. КТ

Программирование и алгоритмические языки, 2019

Содержание лекции

Этапы решения задач на ЭВМ

Решение любой задачи на ЭВМ состоит из нескольких этапов, среди которых основными являются следующие:

1. постановка задачи;
2. формализация (математическая постановка задачи);
3. выбор метода решения;
4. разработка алгоритма;
5. составление программы;
6. отладка программы;
7. вычисление и обработка результатов.

Последовательное выполнение перечисленных этапов составляет полный цикл разработки.

При постановке задачи

первостепенное внимание должно уделяться выяснению конечной цели и выработке общего подхода к исследуемой проблеме. Правильно сформулировать задачу иногда не менее сложно, чем ее решить.

Формализация

сводится к построению математической модели рассматриваемого явления, когда в результате анализа задачи:

- определяются объем и специфика исходных данных;
- вводится система условных обозначений;
- устанавливается принадлежность решаемой задачи к одному из известных классов задач;
- выбирается соответствующий математический аппарат;

Выбор метода решения

необходим после того, как определена математическая формулировка задачи.

- применение любого метода приводит к построению ряда формул и к формулировке правил, определяющих связи между этими формулами;
- все это разбивается на отдельные действия так, чтобы вычислительный процесс мог быть выполнен машиной;

Разработка алгоритма

данный этап заключается в разложении вычислительного процесса на возможные составные части, установлении порядка их следования, описании содержания каждой такой части в той или иной форме.

В процессе разработки алгоритм проходит несколько этапов детализации.

- первоначально составляется укрупненная схема алгоритма, в которой отражаются наиболее важные и существенные связи между исследуемыми процессами.
- на последующих этапах раскрываются выделенные на предыдущих этапах части вычислительного процесса.

Алгоритм

однозначно определенная на некотором языке конечная последовательность предписаний (инструкций, команд), задающая порядок исполнения элементарных операций для систематического решения задачи.

Семь параметров, характеризующих алгоритм:

1. Совокупность возможных исходных данных.
2. Совокупность возможных результатов.
3. Совокупность возможных промежуточных результатов.
4. Правило начала процесса обработки данных.
5. Правило непосредственной обработки.
6. Правило окончания обработки.
7. Правило извлечения результата.

Алгоритм должен обладать свойствами: массовость, результативность.

Средства записи алгоритмов

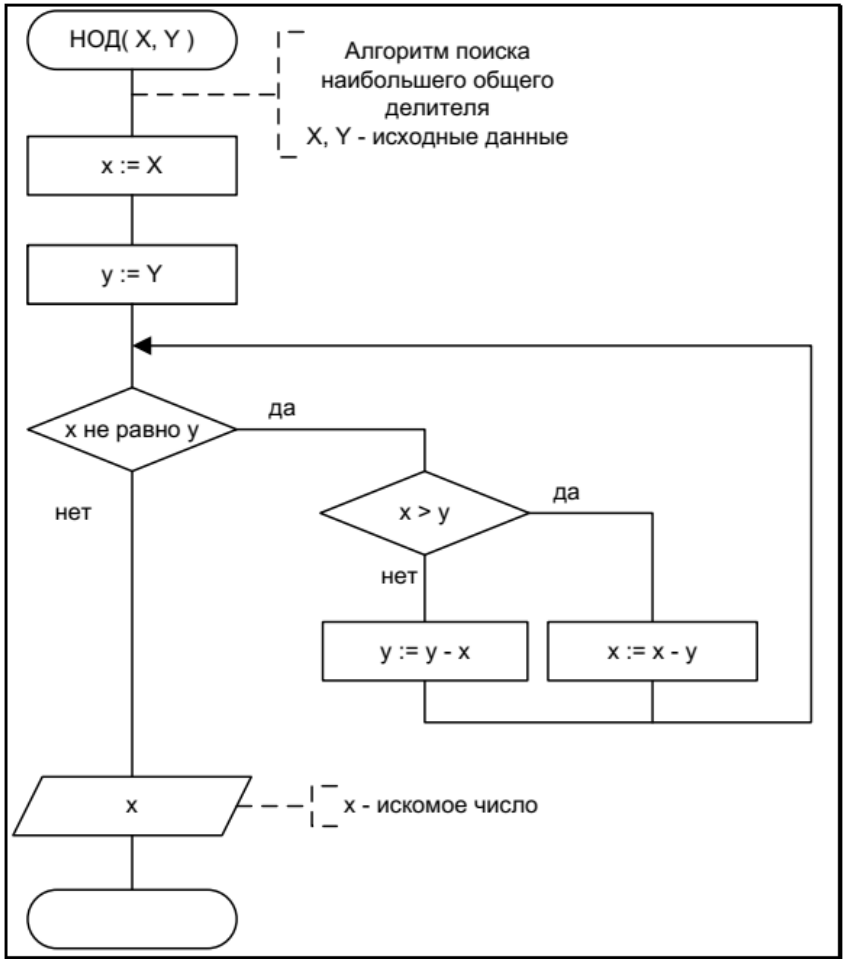
1. текстуальная форма записи;
2. схемы алгоритмов;
3. диаграммы Нэсси-Шнейдермана;
4. диаграммы Дейкстры;
5. псевдокод;
6. запись в форме программы на языке программирования;
7. ДРАКОН-схемы;
8. Р-схемы

Текстуальная форма записи алгоритма

1. Скопировать значение X во вспомогательную переменную x.
2. Скопировать значение Y во вспомогательную переменную y.
3. Если $x \neq y$, перейти к п. 4, иначе – к п. 7.
4. Если $x > y$, перейти к п. 5, иначе – к п. 6.
5. Записать в x результат вычисления выражения $x-y$ и перейти к п. 3.
6. Записать в y результат вычисления выражения $y-x$ и перейти к п. 3.
7. Конец. x – результат работы.

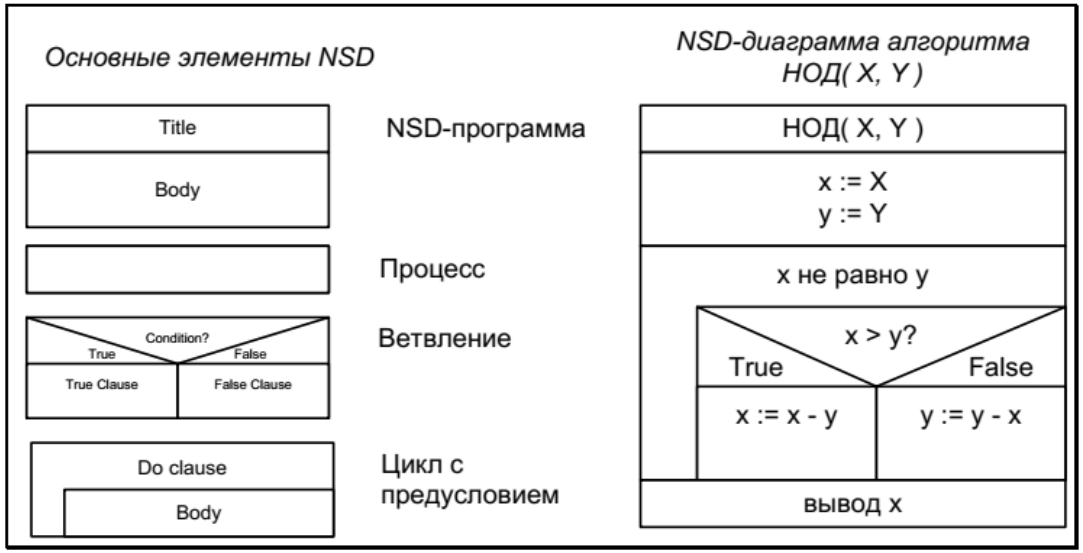
- \+ наименее формализованная
- \- не обеспечивается наглядность решения

Блок-схема алгоритма



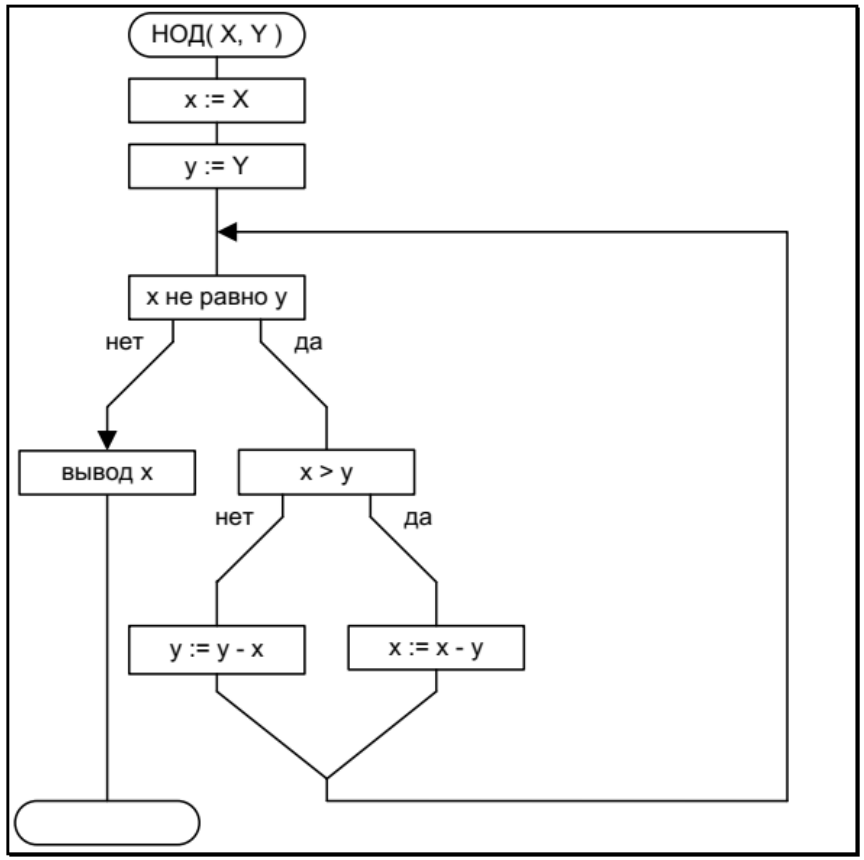
- \+ регулируются стандартом ISO 5807-85 и российским [ГОСТ 19.701-90]
- \- обеспечивают наглядность только для простых алгоритмов

Диаграммы Нэсси-Шнейдермана



- \+ менее запутанные, чем текст и блок-схемы
- \+ методически удобны при обсуждении областей видимости программных объектов
- \- для сложных программ либо внешний прямоугольник будет очень большим, либо внутренние элементы – слишком маленькими

Диаграммы Дейкстры



- \+ "более ограниченная"топология, чем текст и блок-схемы
- \- не был востребованы разработчиками стандартов на схемы алгоритмов и программ

Псевдокод

```
НОД( X, Y )
    x:=X;
    y:=Y;
    пока( x ≠ y ) повторять
        если( x > y ) то x:=x-y;
        иначе           y:=y-x;
    конец цикла
    вывести x
конец
```

- + эффективная форма текстуальной записи алгоритма, абстрагированная от конкретного языка
- + используется, когда программисты хотят отложить окончательную запись решения на более позднее время

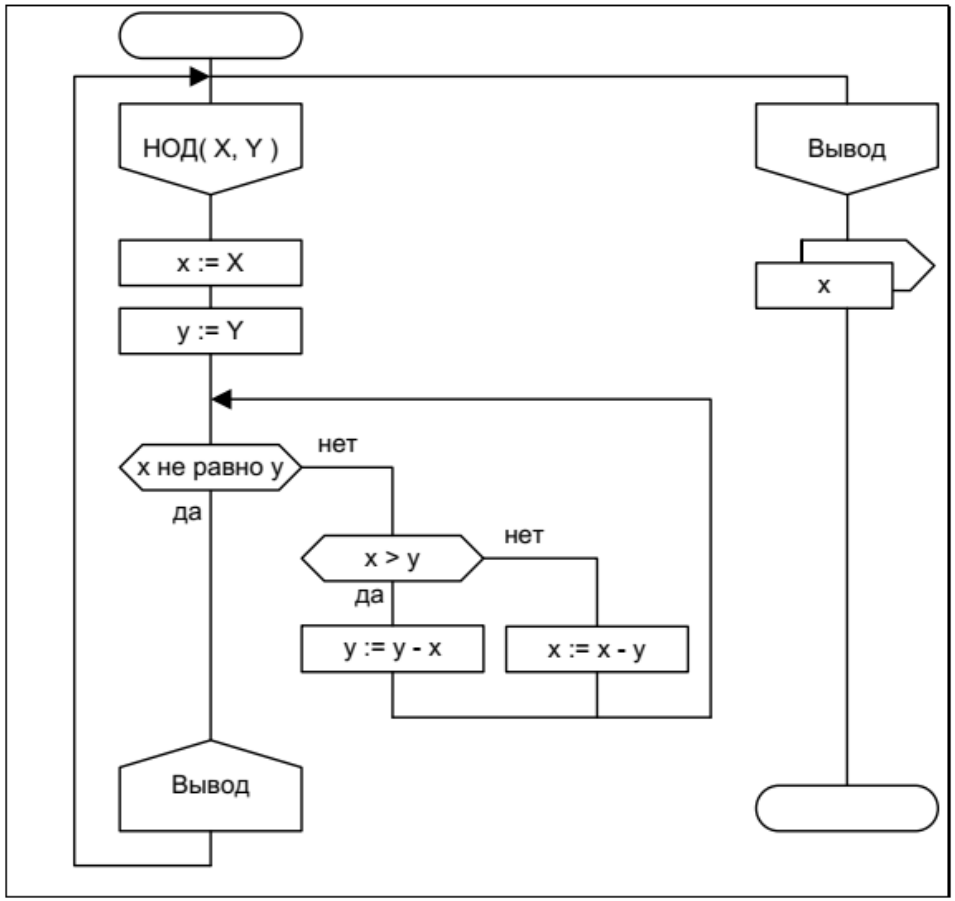
Запись на языке C

```c
int grtCmnDivsor( int X, int Y )
{
    int x = X;
    int y = Y;
    while( x != y )
    {
        if( x > y ) x = x - y;
        else        y = y - x;
    }
    return x;
}
```

Запись на языке Pascal

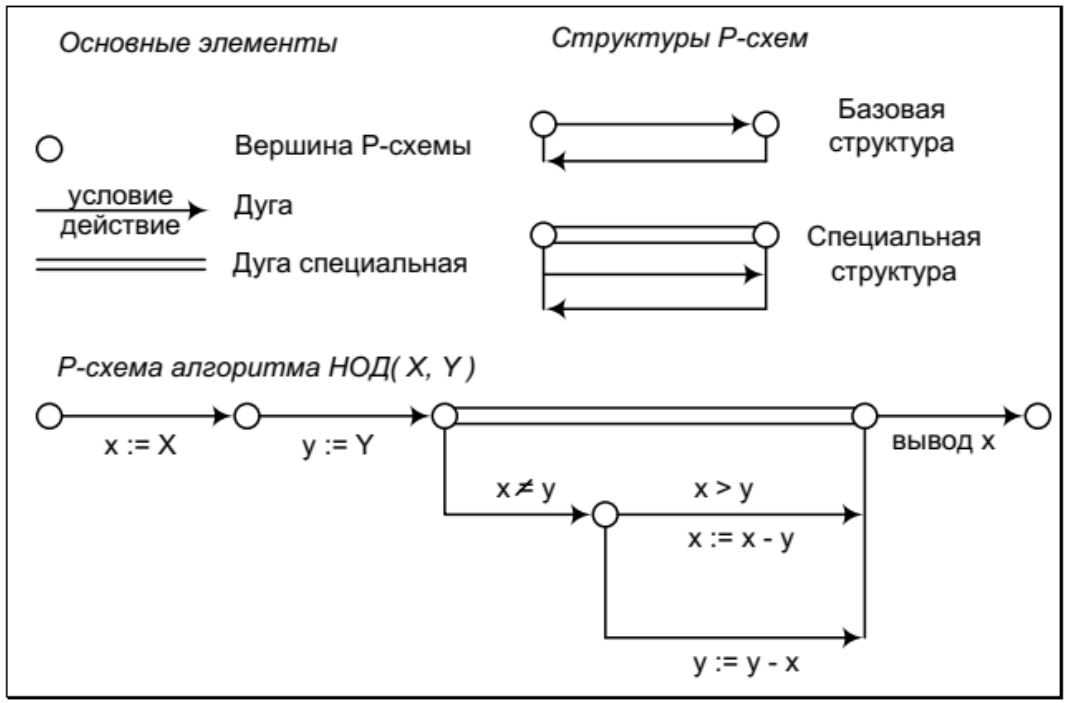
```pascal
function GrtCommonDivider(x, y: integer): integer;
begin
  while x <> y do
    if x > y then
      x := x - y
    else
      y := y - x;
  GrtCommonDivider := x
end;
```

Запись на визуальном языке ДРАКОН



Построен на основе визуального языка ДРАКОН и на основе шампур-метода как абстрактной визуальной модели программы.

P-схемы



P-схемы являются центральным элементом визуального ядра P-технологии, разработанной в институте кибернетики Академии наук Украины.

Алфавит

набор символов (литер), используемых для записи программ.

Из литер конструируются лексемы (tokens) – относительно обособленные элементы синтаксических конструкций языка. Наиболее распространены пять основных категорий лексем:

- идентификаторы – для записи символических имен программных объектов;
- ключевые слова, которые, вообще говоря, являются специальным классом идентификаторов – для записи инструкций языка;
- литералы: целочисленные, с плавающей точкой, символьные, строковые – для записи констант;
- знаки операций (арифметических, логических, операций отношения и др.) – для записи элементарных действий над данными;
- знаки пунктуации – однолитерные лексемы, используемые для структурирования и организации программного кода.

Алфавит языка C++

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
0 1 2 3 4 5 6 7 8 9
+ - * / =
() {} [] <>
' " ! # ~ % ^ & _ ; : , . ? \ |
Символы пробелов и табуляции
```

Алфавит языка Pascal

26 латинских строчных и
26 латинских прописных букв:
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z

10 цифр:
0 1 2 3 4 5 6 7 8 9

знаки операций:
+ - * / = <> < > <= >= := @

ограничители (разделители):
. , ' ( ) [ ] (. .) { } (* *) .. : ;

подчеркивание _

спецификаторы:
^ # $

Трансляция

перевод программы на компилируемом языке высокого уровня в программу на машинных кодах

При трансляции смысл программы не должен измениться, должна измениться только форма ее представления, так как правила составления предложений языка строго регламентированы синтаксисом и семантикой.

Синтаксис

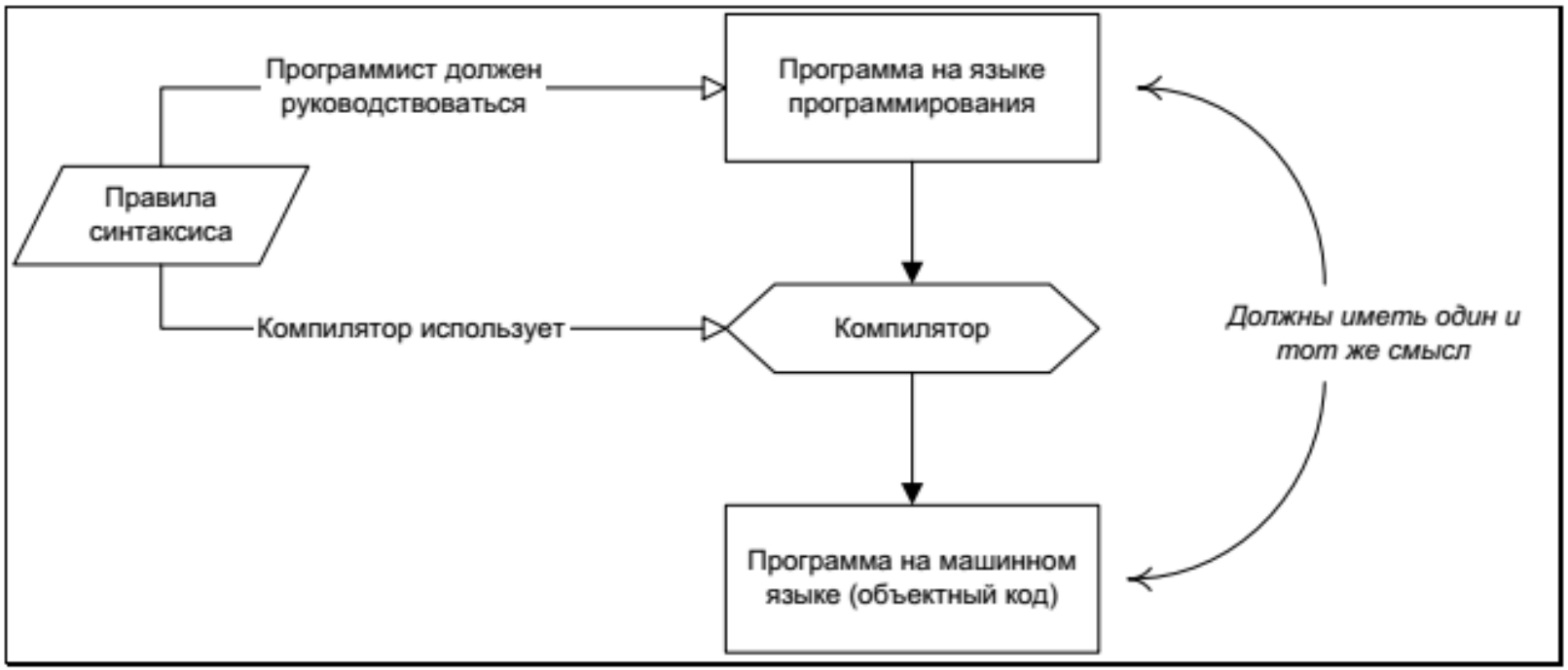
правила построения языка и его конструкций.

Семантика

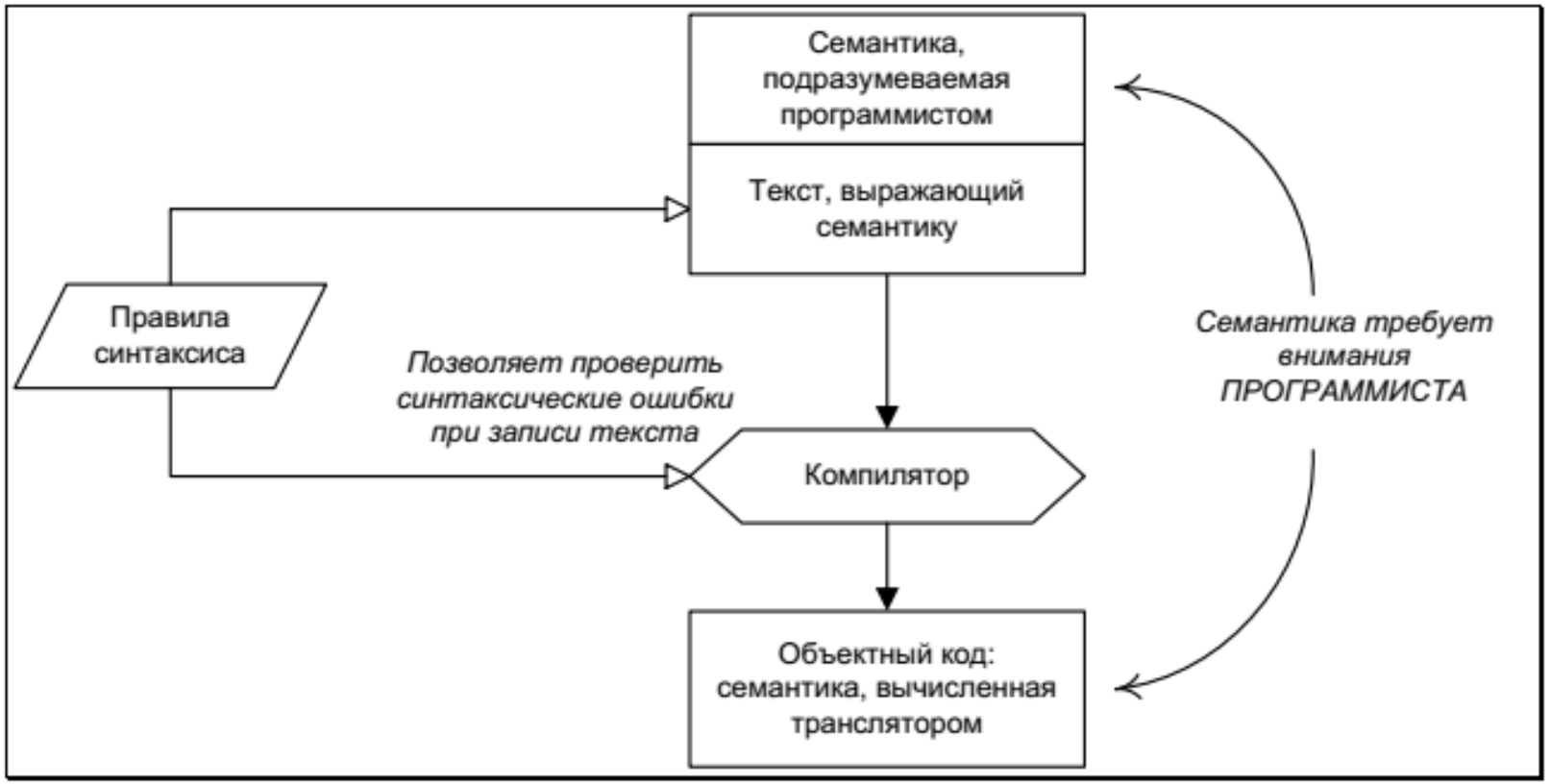
строго определенный смысл каждой конструкции языка.

Синтаксически-ориентированная трансляция

вычисление единственно возможного смысла программного кода на основании анализа синтаксиса.



Синтаксис и семантика

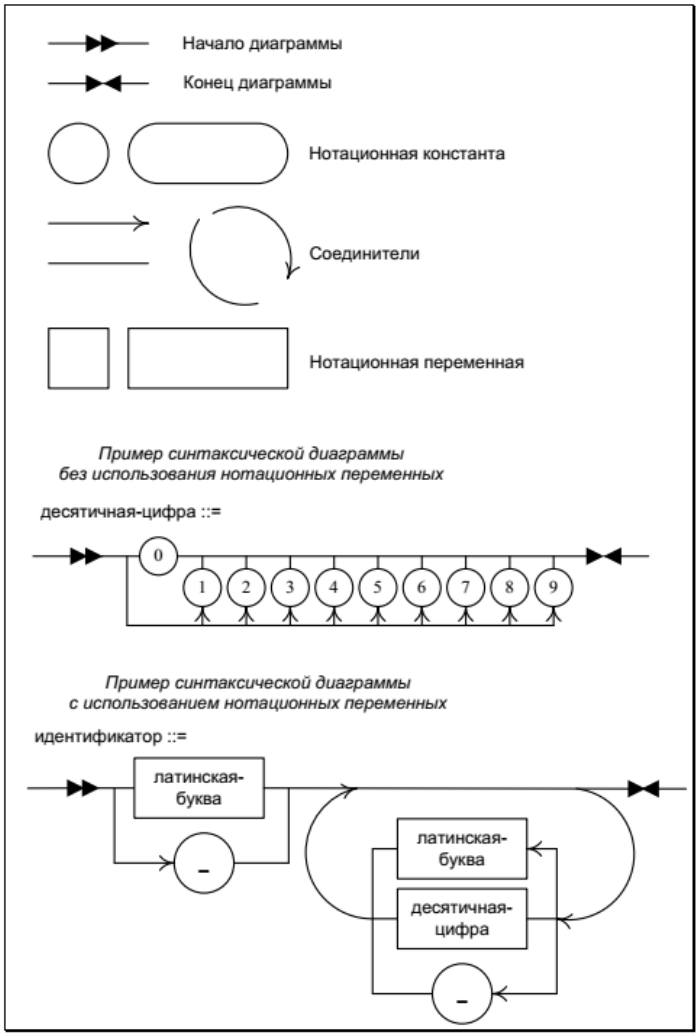


Описание синтаксиса в форме правил Бэкуса-Наура

```
десятичная-цифра ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

идентификатор ::= { <латинская-буква} | '_' } [ <латинская-буква> |
                  <десятичная-цифра> | '_' ]...
```

Синтаксические диаграммы Н. Вирта



Идентификатор

одна из основных синтаксических конструкций, используемых в программах.

- любое имя, используемое в программе, должно быть идентификатором;
- идентификатор может состоять из одного или более символов (латинских букв, цифр, символов подчеркивания);
- первым символом должна быть латинская буква или знак подчеркивания;
- максимальная длина идентификатора зависит от компилятора.

Регситр символов

- не важен для языка Pascal;
- важен для языка C/C++.

- ряд идентификаторов явно зарезервирован для нужд языка. Это – ключевые (служебные) слова языка;
- идентификаторы, начинающиеся с двойного подчеркивания, или с подчеркивания, вслед за которым следует прописная буква, считаются зарезервированными для использования системой (C++).
- в некоторых средах разработки существуют свои стандарты, которым рекомендуется следовать, если предполагается совместное использование программного кода коллективами разработчиков.

Ключевые слова

идентификаторы, имеющие предопределенный смысл в языке.

Ключевые слова языка C

asm	Вставка ассемблерного кода в код на C
auto	Спецификатор класса памяти «автоматическая переменная»
break	Прекращение итераций цикла, выход из мультиветвления
case	Ветвь мультиветвления
char	Спецификатор типа «символьный»
continue	Переход к очередной итерации цикла
default	Ветвь мультиветвления «по умолчанию»
do	Заголовок цикла с постусловием do...while
double	Спецификатор типа «с плавающей точкой двойной точности»
else	Ветвь «иначе» в инструкции if
enum	Спецификатор перечисляемого типа
extern	Спецификатор доступа к внешним объектам
float	Спецификатор типа «с плавающей точкой»
for	Заголовок цикла с предусловием for

Ключевые слова языка Pascal

and	end	nil	set
array	file	not	then
begin	for	of	to
case	function	or	type
const	goto	packed	until
div	if	procedure	var
do	in	program	while
downto	label	record	with
else	mod	repeat	

Правила записи текстов программ могут быть:

- жестко зафиксированы (взаимное расположение конструкций языка жестко регламентировано);
- полуфиксированными (в ассемблерных языках, Python);
- свободным (Pascal, C/C++).

При свободной записи текстов программ:

- синтаксические конструкции могут начинаться с любой позиции строки текста программы;
- одна инструкция может располагаться в нескольких строках программы
- несколько инструкций могут располагаться в одной строке
- между отдельными лексемами может быть помещен один или более пробелов или других символов пробельной группы (табуляции, символы перевода строки)
- поскольку символ конца строки считается символом-разделителем, один идентификатор не может располагаться на нескольких строчках

Рекомендованные правила оформления исходного кода:

- вертикальное форматирование: действия, составляющие один логический блок, комментарии, сопровождающие текст, символы, ограничивающие обособленные блоки (например, фигурные скобки в языке C++, пары ключевых слов begin, end в языке Pascal) записываются друг под другом с выравниванием по левому краю.
- Горизонтальное форматирование: при оформлении перехода к вложенному логическому блоку (в ветвлениях и циклах) используются отступы, или табуляции.
- Оставляются пустые строчки между обособленными логическими фрагментами, определениями функций, структурами данных и т.д.
- Текст организуется таким образом, что в отдельной строке помещается одна высокоуровневая инструкция, за естественным исключением тех случаев, когда группа инструкций образует единое целое в синтаксическом или в математическом смысле.

Комментарии

инструмент сопровождения программного кода.

Комментарии языка C/C++

```
/* это - комментарий Си */
/* комментарий Си может
располагаться
на нескольких строчках и заканчивается в любом месте строки */

// Это - однострочный комментарий, продолжающийся до конца строки

/* При наличии однострочных комментариев возможно
// вот такое вложение комментария.
Это иногда удобно при отладке */
```

Комментарии языка Pascal

```
(*
   Многострочный комментарий
   языка Pascal
*)

//однострочный комментарий FreePascal

{
   Второй вид многострочного комментария
   языка Pascal
}
```

Виды комментариев с точки зрения их назначения:

- **заголовочные, или титульные, комментарии** - комментарии в начале файла (модуля), содержащие, в частности, информацию об имени файла (и, возможно, расшифровку этого имени).
- **выделяющие, или предваряющие, комментарии** - размещаются перед относительно обособленными фрагментами текста, служат промежуточными заголовками к группам данных или действий или объясняют назначение отдельных подпрограмм.
- **сопровождающие, или поясняющие, комментарии** - обычно располагаются справа от основного текста и предназначены не только для пояснения отдельных моментов, но и просто облегчают читателю (в том числе и самому проектировщику) ориентирование в тексте.
- **констатирующие комментарии** - подводят итог некоторого фрагмента, указывают на состояние программы или каких-либо данных по достижении этой точки программы.