Информатика

Алгоритмическое обеспечение информационных технологий

Средства информационных технологий

Содержание

- Понятие алгоритмического обеспечения
- Алгоритм
- Язык программирования
- Техника разработки программ

4

Алгоритмическое обеспечение

Алгоритмическое обеспечение (brainware), или просто *алгоритмы* – точно сформулированные правила, определяющие процесс преобразования информации для решения поставленной задачи.

Дано:
a, *b*, *c* – вещественные
Найти:
корни уравнения $ax^2+bx+c=0$

```
Алгоритм КОРНИ_УРАВНЕНИЯ;
вход вещ a, b, c;
выход вещ x1, x2;
перем вещ D;
начало
  D:=b*b-4*a*c;
  если D<0 то
    аварвыход("Корней нет");
  иначе
    x1:=(-b+sqrt(D))/(2*a);
    x2:=(-b-sqrt(D))/(2*a);
  конец если
конец
```

5

Алгоритм

Алгоритм – это конечный набор правил, который определяет последовательность операций для решения конкретного множества задач и обладает следующими важными чертами: конечность, определённость, эффективность.

[Дональд Кнут. Искусство программирования, том 1. Основные алгоритмы, 2006 г.]

6

Ал-Хорезми

- *Абу-Джафар Мухаммед ибн Муса ал-Хорезми (محمد بن موسى الخوارزميّ المجوسيّ القطربّلي) (783-850 гг.)* – арабский математик.
- Главные труды:
  - *«Китаб ал-Джебр»* – общие правила решения арифметических задач при помощи уравнений;
  - *«Об индийском счете»* – правила записи натуральных чисел с помощью арабских цифр и правила действий над ними «столбиком».
  - Арабское название его «Книги о сложении и вычитании» было в XII веке переведено на латынь как Algoritmi de numero Indorum («Алгоритми о счёте индийском»).

7

Свойства алгоритма

1. *Дискретность*: алгоритм состоит из конечного числа шагов, которые выполняются в дискретном времени.
2. *Элементарность шагов*: объем работы на любом из шагов алгоритма не превышает некоторой константы, зависящей от характеристик исполнителя алгоритма, но не от входных данных и промежуточных результатов алгоритма.
3. *Определенность (детерминированность)*: результаты выполнения каждого шага алгоритма вычисляются однозначно и не зависят от случайных факторов.
4. *Конечность (завершаемость)*: алгоритм должен приводить к решению задачи за конечное число шагов (либо останавливаться из-за невозможности получить решение).
5. *Массовость*: алгоритм должен быть применим для некоторого класса задач, различающихся лишь исходными данными.

8

Пример алгоритма

```
Алгоритм МАКСИМУМ_В_МАССИВЕ;
вход вещ мас A[1..N];
выход вещ Max;
перем цел i;
начало
  Max:=A[1];
  i:=2;
  пока i<=N
    если Max<A[i] то
       Max:=A[i];
    конец если
    i:=i+1;
  конец пока
конец
```

1. **Дискретность**
   Очевидна
2. **Элементарность шагов**
   Количество операций ≤6
3. **Определенность**
   Очевидна
4. **Конечность**
   Очевидна
5. **Массовость**
   Алгоритм применим для любого натурального значения **N**

9

Языки для записи алгоритма

- *Естественный язык* – алгоритм излагается произвольно.
- *Язык блок-схем* – алгоритм изображается как последовательность связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий.
- *Язык программирования* – формальный язык для записи алгоритмов.

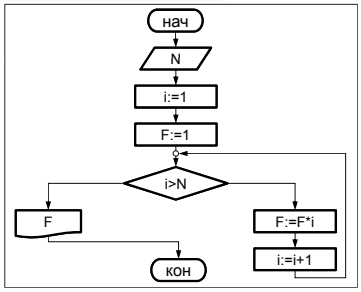
10

Пример записи алгоритма

Возьми само число
И те, что перед ним,
Всё перемножь; тогда
Получишь, что
искал.

f ≡ eq ° [id, 1] → 1;
x ° [id, f ° – ° [id, 1]]

F(n, n*y):–F(m, y), m=n-1
F(1, 1)




```
function Factorial
        (N: Integer): Integer;
var
   i, F: Integer;
begin
   F:=1;
   for i:=1 to N do
       F:=F*i;
   Factorial:=F;
end;
```

11

Язык программирования

- *Язык программирования* – система обозначений для описания программ (алгоритмов и структур данных).
- Наиболее распространенные языки: Pascal, C, C++, Ada, Java, BASIC, FORTRAN, PROLOG, LISP.

12

Задание языка программирования

- *Алфавит* – множество символов, используемых для записи предложений данного языка.
- *Синтаксис* – множество всех предложений, принадлежащих данному языку.
- *Семантика* – смысл предложений языка (что делает компьютер при выполнении предложений данного языка).

13

Пример: язык M17

- **Алфавит**
  { Юстас, Алекс, покупает, продает, шкаф, стол, славянский, немецкий }

- **Синтаксис**
  предложение это **подлежащее сказуемое определение дополнение**
  подлежащее это Алекс или Юстас
  сказуемое это покупает или продает
  определение это славянский или немецкий
  дополнение это шкаф или стол

14

Пример: язык M17

- **Синтаксически верные предложения**
  Юстас продает славянский шкаф
  Алекс покупает немецкий стол
  Юстас покупает немецкий шкаф
- **Синтаксически НЕверные предложения**
  Юстас Алекс продает славянский шкаф
  Юстас покупает стол
  Юстас продает славянский шкаф.
- **Семантика** (для одного предложения)

Центр я на грани провала
Штирлиц

15

Способы задания синтаксиса языка

- Расширенные формулы Бэкуса-Наура (РБНФ)

- Синтаксические диаграммы

- Перечисление всех верных конструкций

16

Построение РБНФ

- *Нетерминальные символы* – заключаются в угловые скобки **<...>** и используются для обозначения синтаксических конструкций языка.
- *Терминальные символы* образуют алфавит языка.
- *Метасимволы*

  ::= есть по определению

  | или

  [...**]** повторение символа 0 или 1 раз

  {...**}** повторение символа произвольное число раз (в т.ч. нуль)

17

Пример: язык М18

- **Алфавит**
  { Юстас, Алекс, изучает, преподает, язык, английский, немецкий, **.** }
- **Синтаксис**
  <предложение> ::= <подлежащее> <сказуемое> [<определение> <дополнение>] **.**
  <подлежащее> ::= Алекс | Юстас
  <сказуемое> ::= изучает | преподает
  <определение> ::= английский | немецкий
  <дополнение> ::= язык
- **Примеры синтаксически верных предложений**
  Алекс изучает. Юстас преподает. Алекс изучает немецкий язык. Юстас преподает английский язык.

18

Пример: рекурсия в РБНФ

- <Целое без знака> ::= <Цифра> |
  <Целое без знака> <Цифра>

  <Цифра> ::= 0 | 1 | 2 | 3 | 4 | 5| 6 | 7 | 8 | 9

- <Двоичное число> ::= <Двоичная цифра> |
  <Двоичное число> <Двоичная цифра>

  <Двоичная цифра> ::= 0 | 1

19

Синтаксические диаграммы

- *Синтаксические диаграммы* – графическое изображение РБНФ.
- Нетерминальные символы

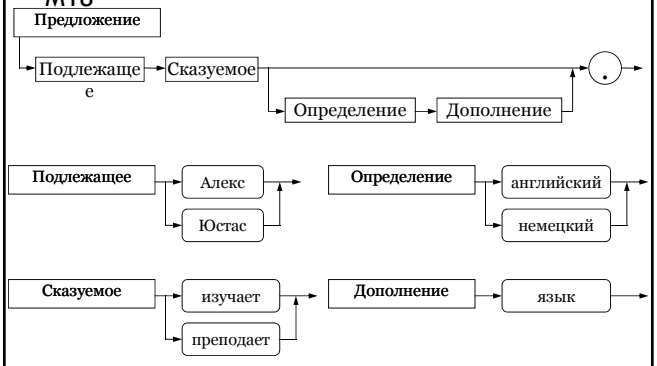
Нетерминальный символ

- Терминальные символы

Терминальный символ

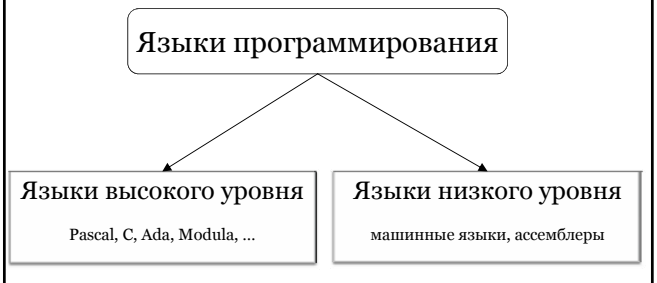
20

Пример: синтаксические диаграммы языка М18



21

Классификация языков программирования



22

Пример: вычисление факториала

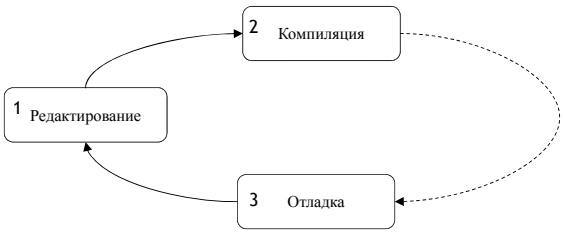
Машинный язык	Язык ассемблера	Pascal	PROLOG

10111010
00001100
00000001
00101110
10001001
00010110
10010001
00000010
10001011
00011110
00101100
00000000
10001110
11011010
10100011 ...
	mov AX,1
	mov BX,N
	@2:
	cmp BX,0
	je @1
	imul BX
	dec BX
	jmp @2
	@1:
		function Fact(N: **Integer**): **Integer**;
		var
		i, F: **Integer**;
		begin
		F:=1;
		for i:=1 **to** N **do**
		F:=F*i;
		Fact:=F;
		end;
			Fact(0,1).
			Fact(1,1).
			Fact(F,N) :-
			N1 **is** N-1,
			N>1,
			Fact(F1,N1),
			F **is** N*F1.

23

Техника разработки программ

- *Технологический цикл разработки программы* на некотором языке программирования представляет собой следующую последовательность действий:



24

Редактирование

- *Редактирование* – процесс написания и исправления исходного текста программы (или ее части) в виде текстового файла.

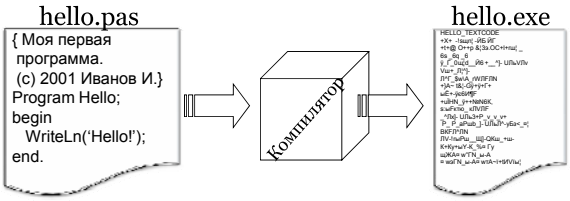
hello.pas

```
{ Моя первая
 программа.
 (c) 2001 Иванов И. }
Program Hello;
begin
  WriteLn('Hello!');
end.
```

25

Компиляция

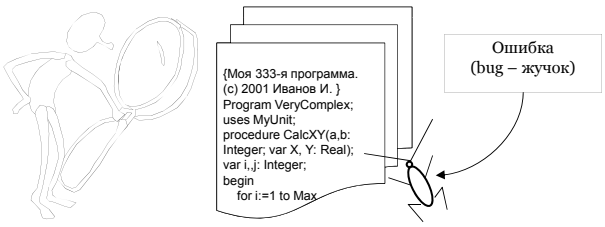
- *Компиляция* – перевод исходного текста программы в машинные коды. Осуществляется специальной программой – *компилятором.*



26

Отладка

- *Отладка* – процесс локализации и исправления ошибок, выявленных во время исполнения программы.



27

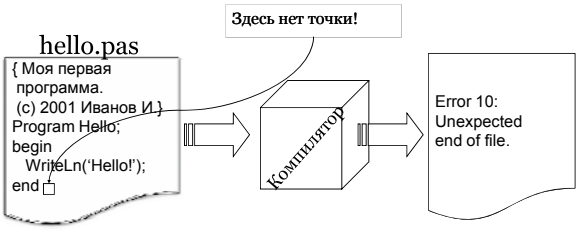
Классификация ошибок

- Синтаксическая ошибка
- Ошибка времени выполнения
- Алгоритмическая ошибка

28

Синтаксическая ошибка

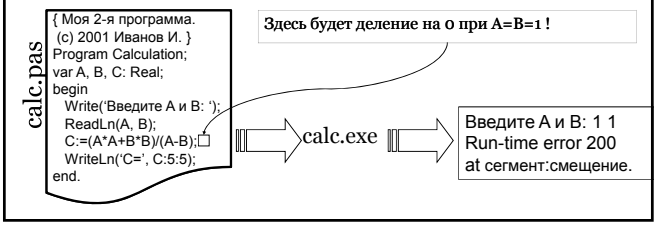
- *Синтаксическая ошибка* – ошибка времени компиляции, происходит при нарушении в тексте программы синтаксических правил языка.



29

Ошибка времени выполнения

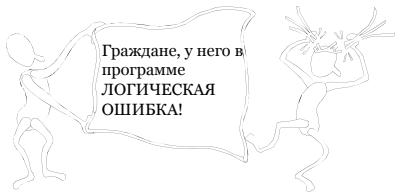
- *Ошибка времени выполнения* происходит при выполнении откомпилированной программы, когда она производит какое-либо недопустимое действие, например, выполняет деление на ноль.



30

Алгоритмическая ошибка

- *Алгоритмическая (логическая) ошибка* – ошибка разработки и кодирования (написания) алгоритма. Программа не содержит ни синтаксических ошибок, ни ошибок времени выполнения, но делает не то, что хотел автор программы.



31

Заключение

- **Алгоритмическое обеспечение (brainware)** – точно сформулированные правила, определяющие процесс преобразования информации для решения различных задач.
- Формально **алгоритм** – это конечный набор правил, который определяет последовательность операций для решения конкретного множества задач и обладает пятью важными чертами: конечность, определённость, ввод, вывод, эффективность.
- Основные свойства алгоритма – **дискретность**, **элементарность шагов**, **детерминированность**, **конечность**, **массовость**.

32

Заключение (2)

- Основные средства записи алгоритма – **естественный язык**, **язык блок-схем**, **язык программирования**.
- **Язык программирования** – это система обозначений для описания программ, которая задается описанием **алфавита**, **синтаксиса** и **семантики**. Синтаксис языка программирования можно задать **расширенными формулами Бэкуса-Наура** или **синтаксическими диаграммами**.
- Языки программирования делятся на **языки низкого** и **высокого уровня (ЯВУ)**. ЯВУ поддерживают понятие **типа данных**, **подпрограммы**, обеспечивают **развитые управляющие структуры**, **мобильность** программ и имеют синтаксис, близкий к естественному языку.

33

Заключение (3)

- Технологический цикл разработки программы на некотором языке программирования включает в себя этапы **редактирования**, **компиляции** и **отладки**.
- Программа на некотором языке программирования может содержать **синтаксические** ошибки, ошибки **времени выполнения** и **логические** ошибки.