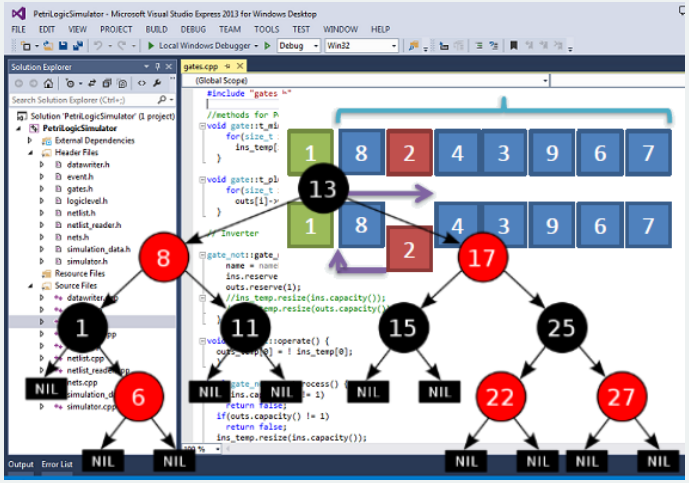
МИЭТ
Национальный исследовательский университет «МИЭТ»
Кафедра ПКИМС

Теория алгоритмов



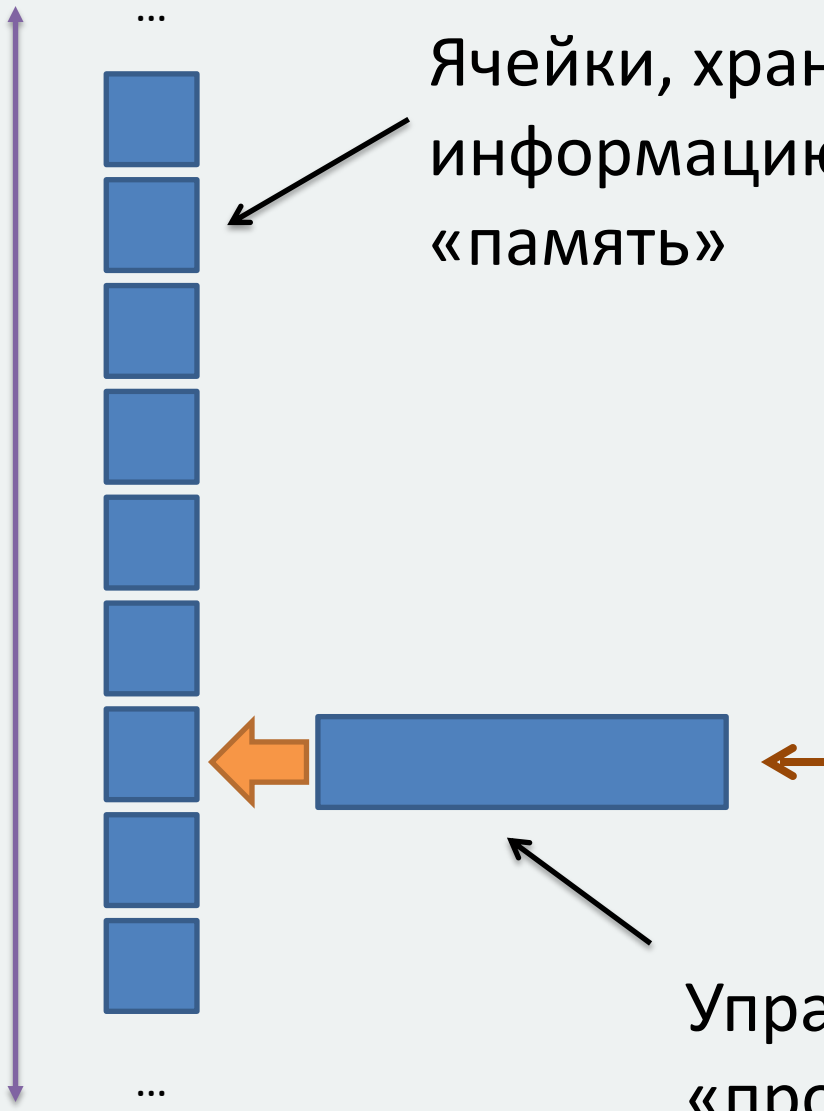
Лекция 1

Общие сведения об алгоритмах

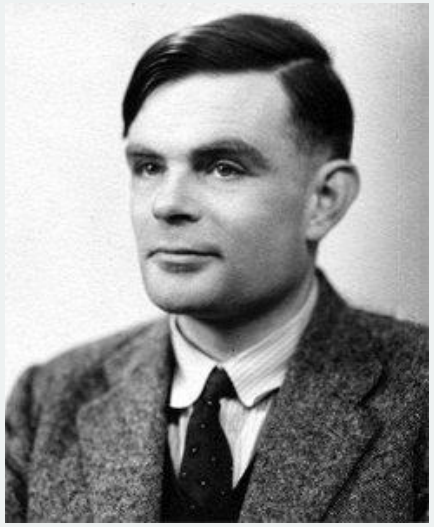
Алгоритм: определение

Алгоритм - это организованная последовательность действий, понятных для некоторого исполнителя, ведущая **за конечное число шагов** к решению поставленной задачи.

Машина Тьюринга



Ячейки, хранящие информацию «память»



Алан Тьюринг

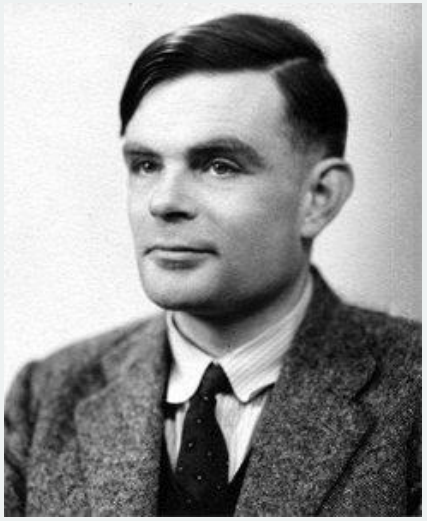
Перечень состояний

Правила перехода

Управляющая машина «процессор»

Тезисы Чёрча-Тьюринга

Любая функция, которая может быть вычислена физическим устройством, может быть вычислена машиной Тьюринга



Алан Тьюринг

Алонзо Чёрч

Любой конечный физический процесс, не использующий аппарат, связанный с непрерывностью и бесконечностью, может быть вычислен физическим устройством

Свойства алгоритмов

- **Дискретность**
- **Корректность**
- **Детерминированность**
- Массовость
- Конечность
- Результативность

Свойства алгоритмов: дискретность

***Дискретность* -** означает, что алгоритм состоит из последовательности отдельных шагов - элементарных действий, выполнение которых на заданном уровне абстракции не представляет сложности и вполне понятно.

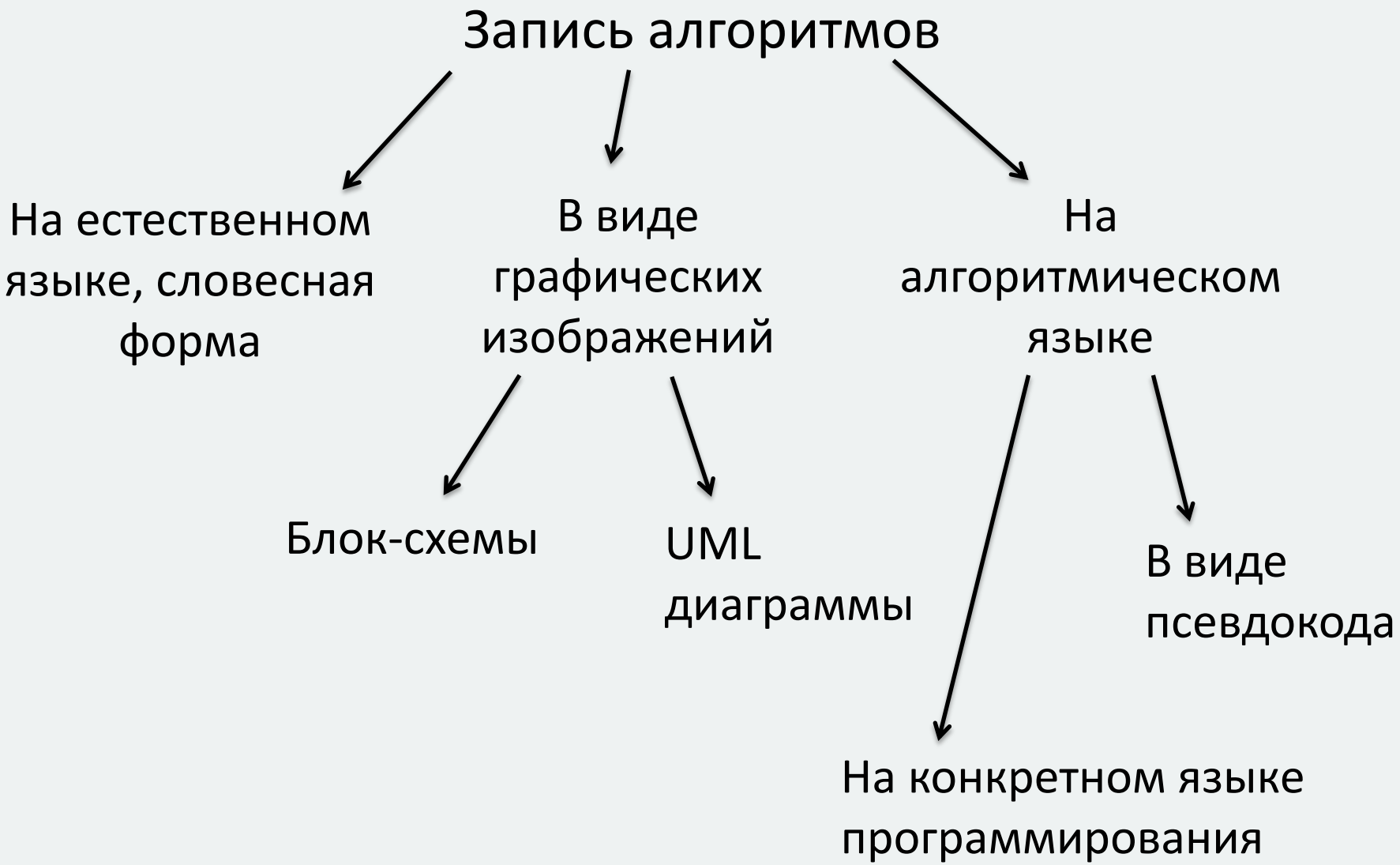
Свойства алгоритмов: корректность

Корректность - означает, что, если алгоритм создан для решения определенной задачи, то для всех исходных данных он должен всегда давать правильный результат и ни для каких исходных данных не будет получен неправильный результат.

Свойства алгоритмов: детерминированность

Детерминированность (*определенность, точность, однозначность*). Это свойство заключается в том, что при задании одних и тех же исходных данных несколько раз алгоритм будет выполняться абсолютно одинаково и всегда будет получен один и тот же результат.

Формы записи алгоритмов

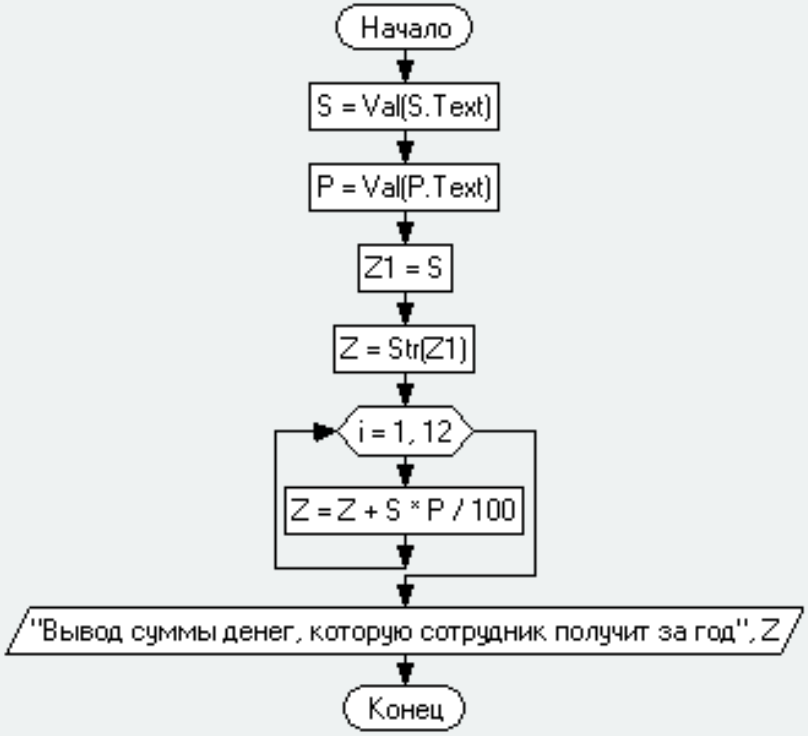


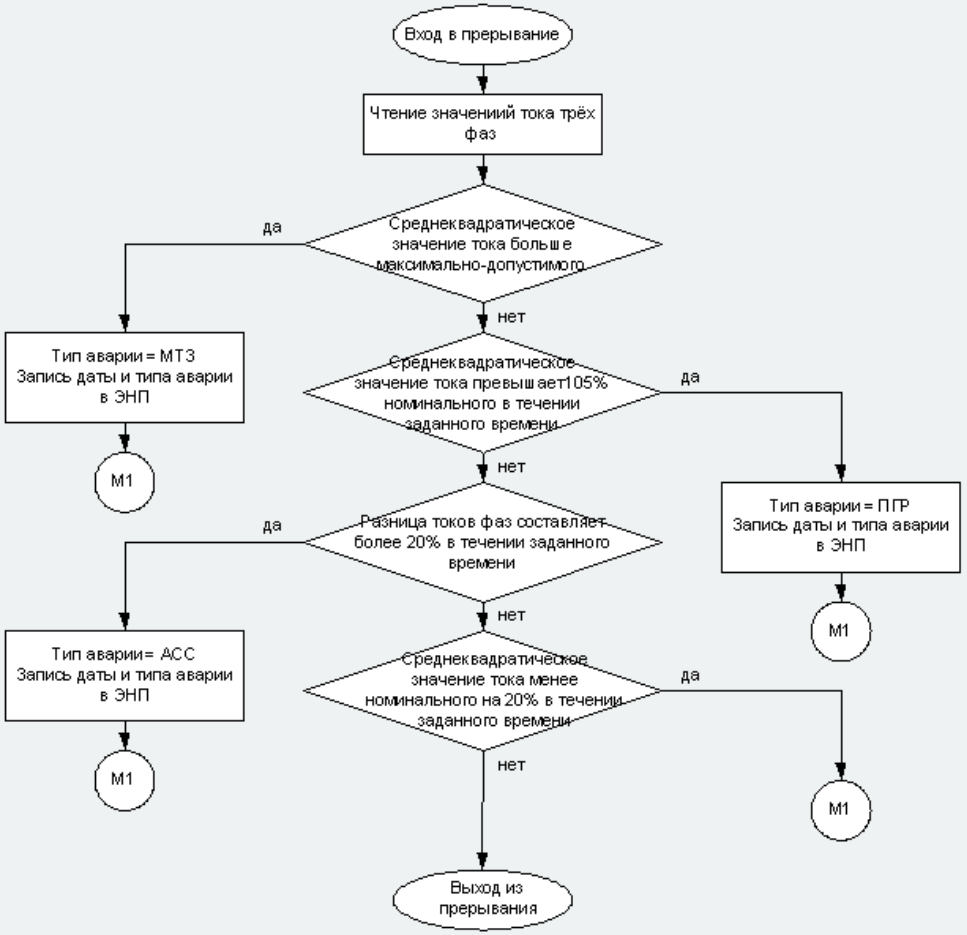
Формы записи алгоритмов: словесная

Программа сложения двух положительных чисел.

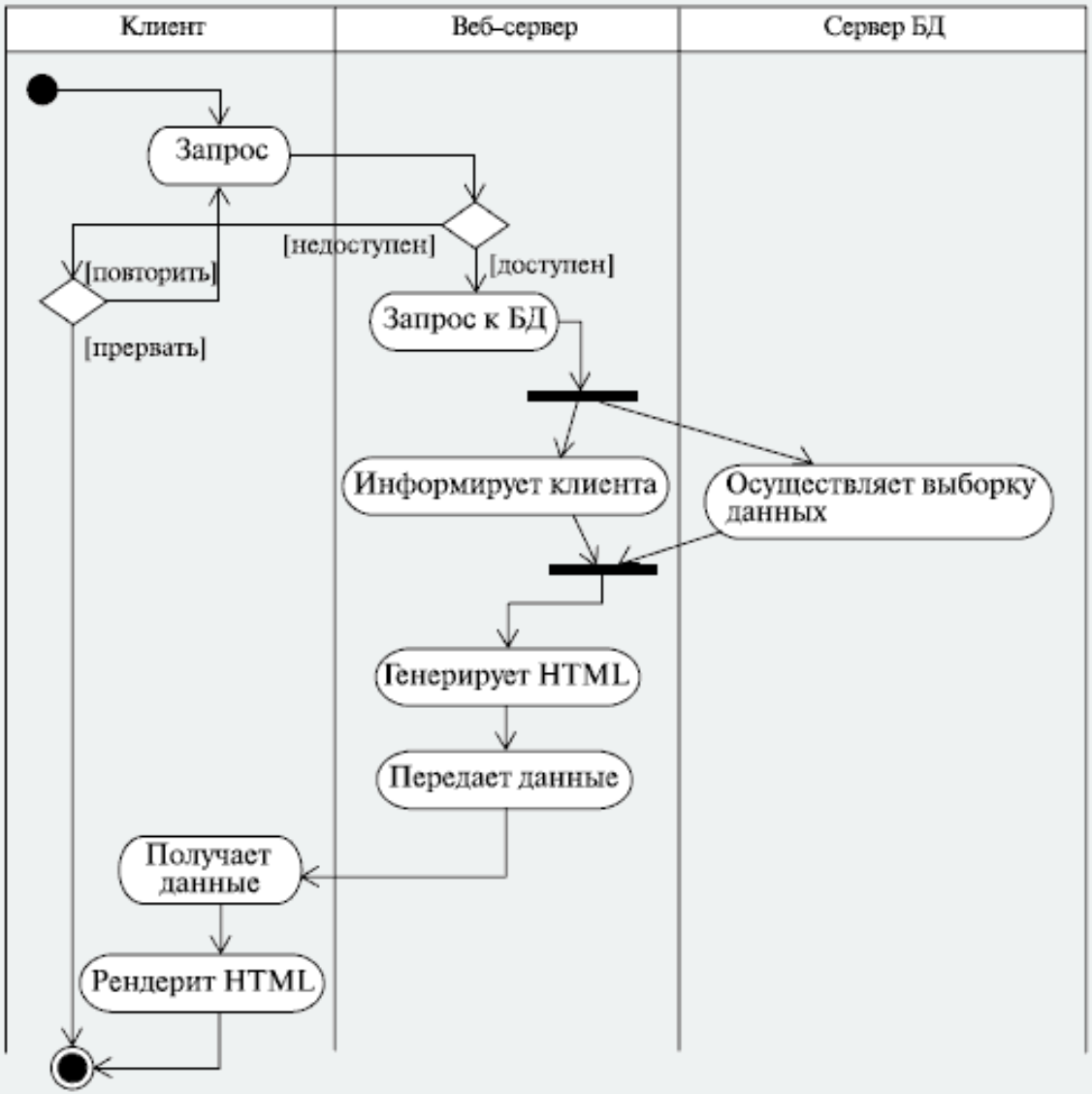
Сначала считываются с клавиатуры два числа, затем выполняется проверка того, что эти числа положительны. Если хотя бы одно число не положительно, выводится соответствующее сообщение и выполняется выход из программы. Если все числа …

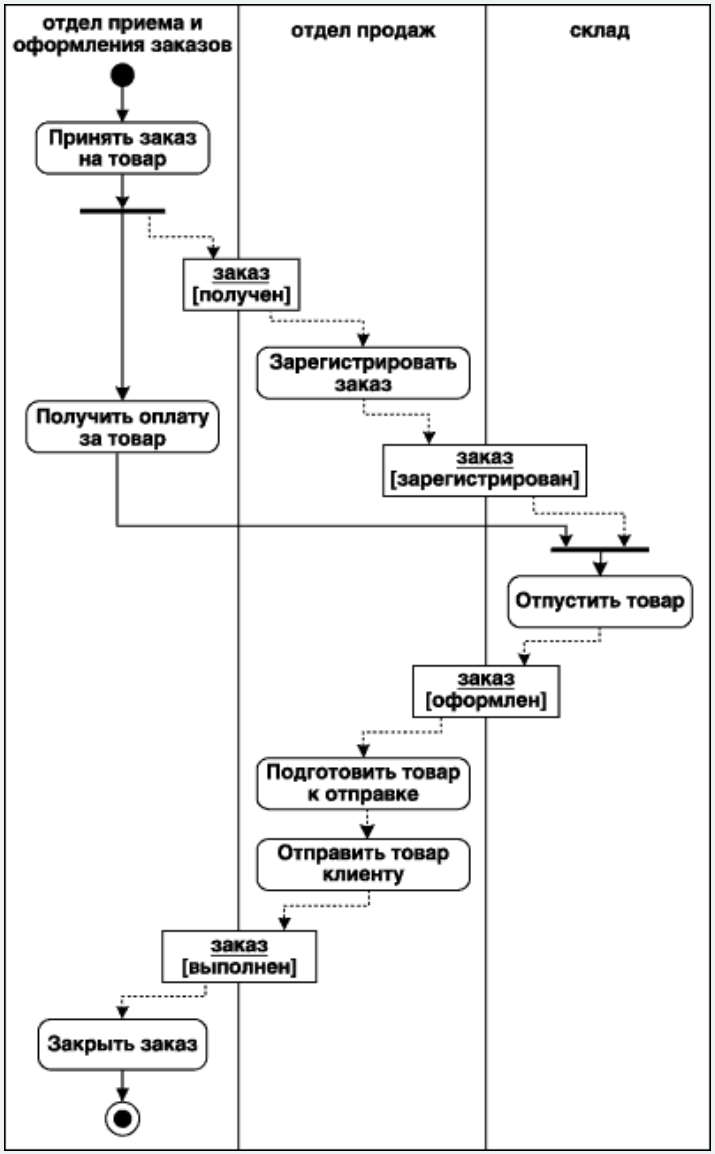
Формы записи алгоритмов: графическая (1) (блок-схемы)





Формы записи алгоритмов: графическая (2) (UML)





Формы записи алгоритмов: код на языке программирования

```
Program diag;
uses crt;

var
  a, b: word;

Function Nod(x, y: word): word;
Begin
  If (x <> 0) then
    Nod := Nod(y mod x, x)
  else
    Nod := y;
End;

BEGIN

  ClrScr;
  writeln('Введите длины сторон прямоугольника: ');
  write('a = ');
  Readln(a);
  write('b = ');
  Readln(b);
  writeln;
  Writeln('Количество единичных квадратов: ', a + b - NOD(a, b));
  Readkey;
END.
```

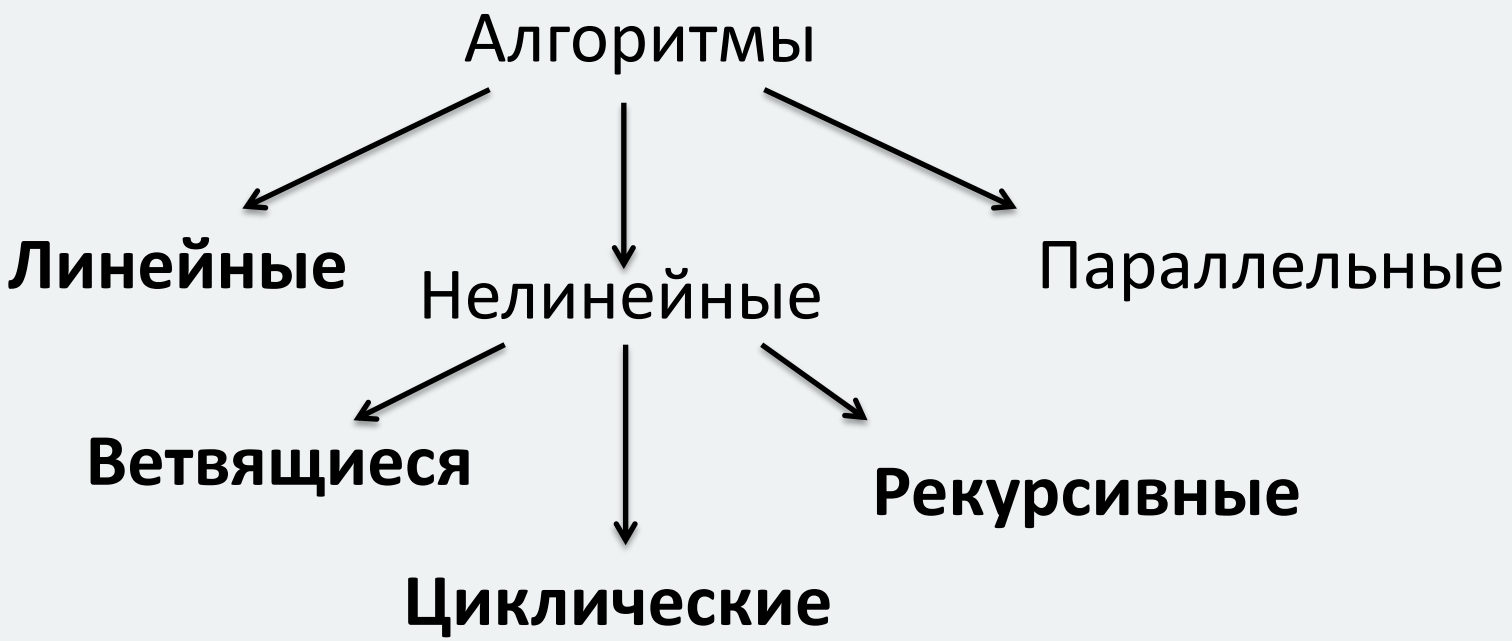
Формы записи алгоритмов: запись в виде псевдокода

```
for (int i = 0; i < N; ++i) {
  scanf("%d", &x);
  printf("%d", x + 2);
}
```

```
for i : 0 → N {
  read x
  print x + 2
}
```

Виды алгоритмов с точки зрения их исполнения

Не рассматриваем вопросы области применения алгоритмов: сортировки, поиска, обхода графа и т.д.
Только виды самих алгоритмов по принципу хода исполнения.



Виды алгоритмов по принципу работы

Детерминированные (жёсткие) алгоритмы – предполагают одну и ту же последовательность действий вне зависимости от входных данных.

Вероятностные (гибкие, адаптивные) алгоритмы – предполагают различное решение в зависимости от некоторых случайным образом генерируемых данных.

Эвристические (статистически верные) алгоритмы – не имеют чёткого обоснования, но дают верный ответ в большинстве случаев.

Парадигмы разработки алгоритмов

Линейные

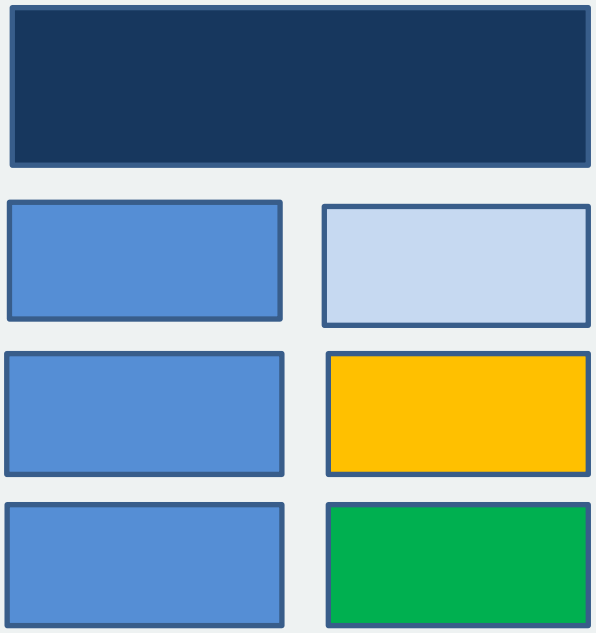
Разделяй и властвуй

Алгоритмы динамического программирования

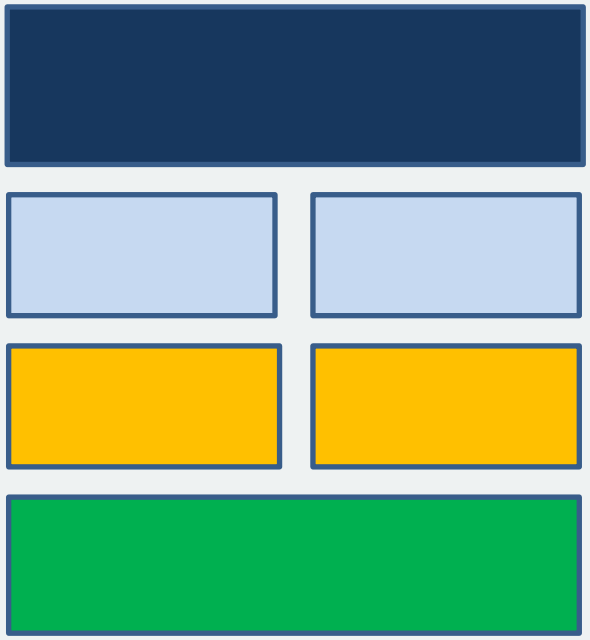
Алгоритмы работающие в ширину и в глубину

Парадигма «Разделяй и властвуй»

Вариант 1



Вариант 2



Основные этапы:

1. разделить целостные данные на части;
2. обработать выбранный/каждый блок данных;
3. при необходимости – объединить результаты обработки разных блоков;

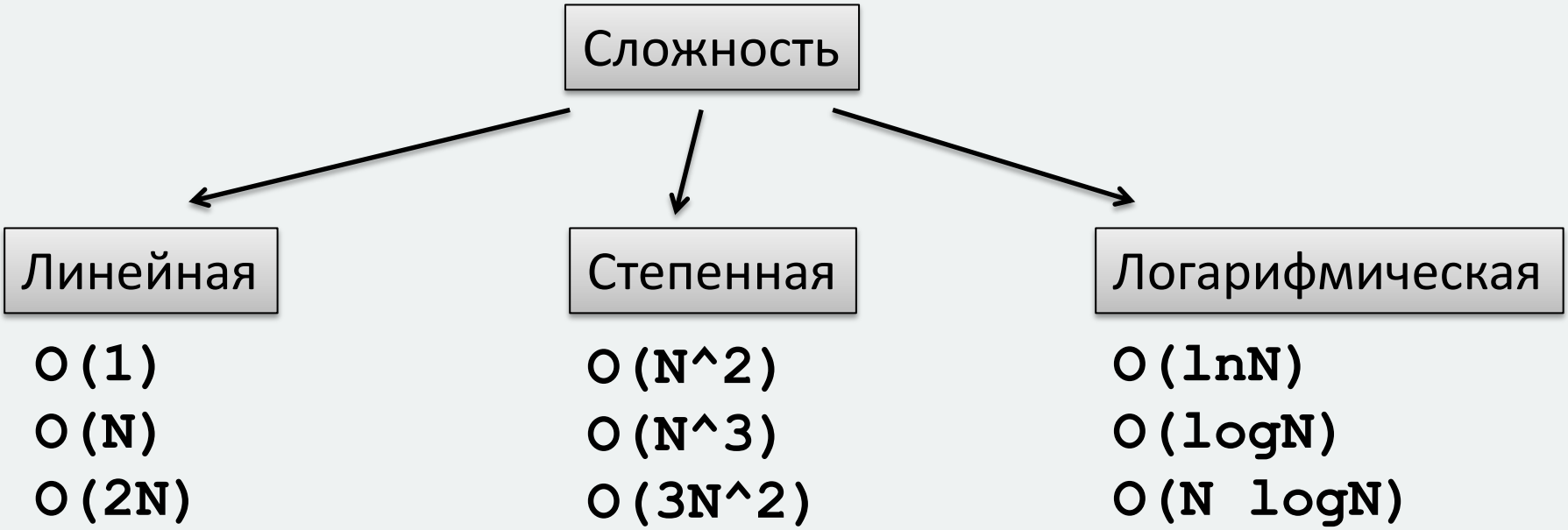
Парадигма динамического программирования (1)

Динамическая парадигма предполагает сохранение результатов предыдущих вычислений.

В основном используется в задачах, где выполняется многократная обработка одинаковых данных.

Чем определяется сложность алгоритма?

Сложность алгоритма – число некоторых операций в зависимости от числа элементов



O(1)
O(N)
O(2N)

сложность доступа к элементу массива
O(1)

сложность доступа к элементу списка
O(N)

O(N^2)
O(N^3)
O(3N^2)

сложность обхода элементов матрицы
O(N^2)

сложность плохих алгоритмов сортировки
O(N^2)

O(lnN)
O(logN)
O(N logN)

сложность хороших алгоритмов поиска и сортировки
O(lnN)
O(logN)
O(N logN)

Демонстрация сложности алгоритмов (1)

Число	N^2	N^3	lnN	log10(N)	Nlog10(N)
1	1	1	0	0	0
2	4	8	0,693147	0,30103	0,60206
3	9	27	1,098612	0,477121	1,431364
4	16	64	1,386294	0,60206	2,40824
5	25	125	1,609438	0,69897	3,49485
6	36	216	1,791759	0,778151	4,668908
7	49	343	1,94591	0,845098	5,915686
8	64	512	2,079442	0,90309	7,22472
9	81	729	2,197225	0,954243	8,588183
10	100	1000	2,302585	1	10
11	121	1331	2,397895	1,041393	11,45532
12	144	1728	2,484907	1,079181	12,95017
13	169	2197	2,564949	1,113943	14,48126
14	196	2744	2,639057	1,146128	16,04579
15	225	3375	2,70805	1,176091	17,64137
16	256	4096	2,772589	1,20412	19,26592
17	289	4913	2,833213	1,230449	20,91763
18	324	5832	2,890372	1,255273	22,59491
19	361	6859	2,944439	1,278754	24,29632
20	400	8000	2,995732	1,30103	26,0206