ГУАП Государственный университет аэрокосмического приборостроения

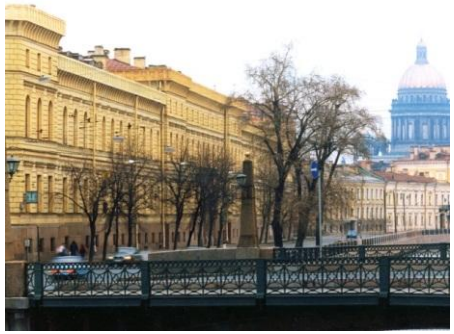
www.guap.ru

Информатика

Рождественская Ксения Николаевна

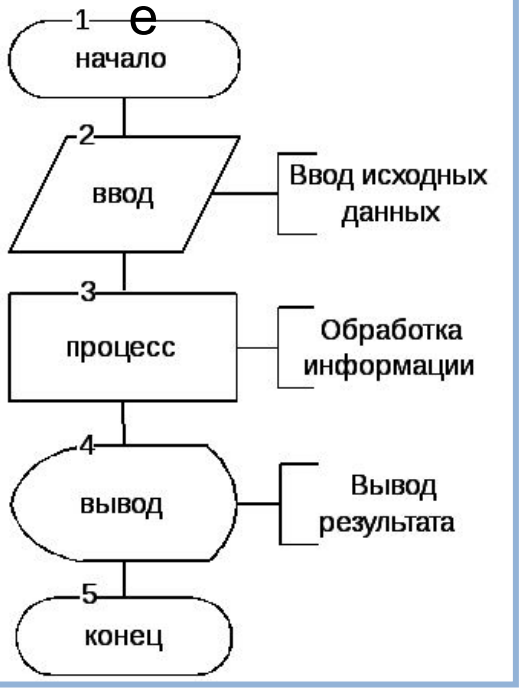
Кафедра 14

ksu.khramenkova@gmail.com

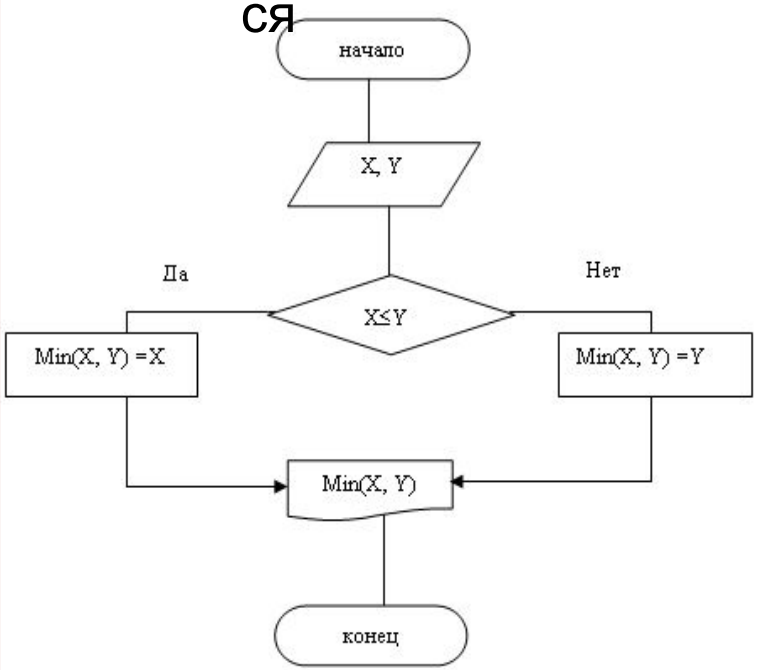


Типы вычислительных процессов

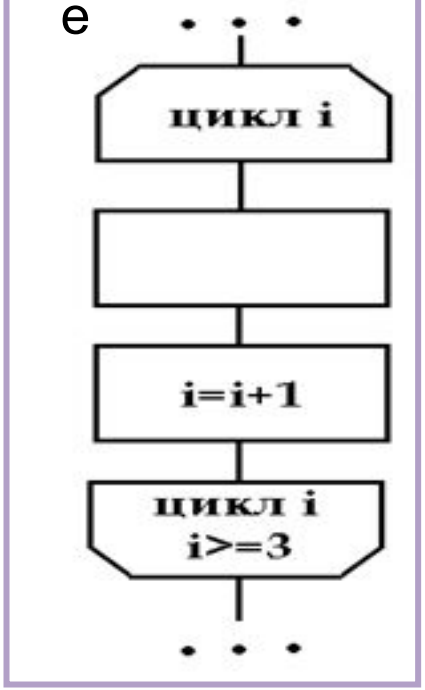
Линейные



Ветвящиеся



Циклические



Управляющие базовые структуры

- линейной композиции
- выбора решения
- повторения

Типы вычислительных процессов

Линейные процессы

- это такой вычислительный процесс, в котором самостоятельные (отдельные) этапы вычислений выполняются в **линейной последовательности** их записи, т.е. в естественно порядке

Блоки размещаются сверху вниз в порядке их выполнения

Типы вычислительных процессов

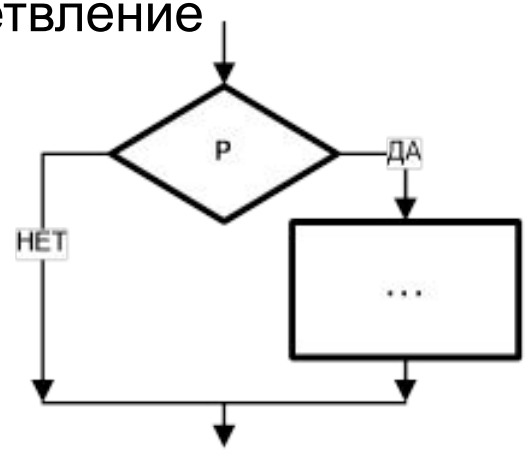
Разветвляющиеся процессы

Разветвляющимся называется такой алгоритм, в котором выбирается **один** из нескольких возможных вариантов вычислительного процесса.

Каждый подобный путь называется **ветвью алгоритма**.

Двустороннее ветвление

Одностороннее ветвление



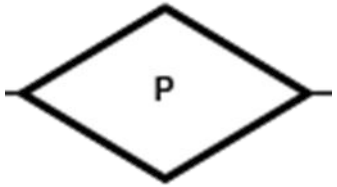
Типы вычислительных процессов

Разветвляющиеся процессы. Особенности обработки разветвляющихся процессов

1. Чтобы **из двусторонней** развилки сделать **одностороннюю**, надо предписание одной из веток (любой) выполнить до условного оператора – это необходимо предпринимать при реализации программ на языках низкого уровня.

2. Для того чтобы в условном операторе **поменять ветки местами**, **условие** надо **поменять на противоположное**.

Операции, которые используются в предикатах: **операции отношения** и **логические** операции

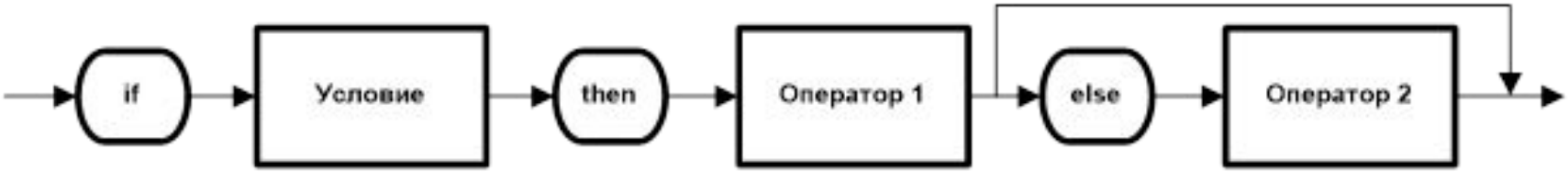


Операции отношения:
=, !=, < =, > = , <, >

Отрицание операций отношения:
=, < >, < =, > = , <, >
!=, = , >, < , >=, <=

Типы вычислительных процессов

Оператор условия If



Может принимать одну из следующих форм

Полный оператор

if «условие» then «оператор 1»
else «оператор 2»

Неполный оператор

if «условие» then «оператор 1»

Типы вычислительных процессов

Оператор условия If

- «оператор 1» и «оператор 2» тоже могут быть условными операторами: (на количество вложений нет ограничений)
  - В неполный оператор можно вложить неполный (а) и полный (б);
  - В полный оператор также можно вложить неполный (в) и

а)	б)	в)

if «условие 1» then
if «условие 2» then
«оператор 1»
ИЛИ
if («условие 1» and «условие 2») then
«оператор 1»
	if «условие 1» then
	if «условие 2» then
	«оператор 1»
	else «оператор 2»
		if «условие 1» then
		if «условие 2» then
		«оператор 1» else «оператор 2»

ВНИМАНИЕ!

else ВСЕГДА ОТНОСИТСЯ К ПОСЛЕДНЕМУ if

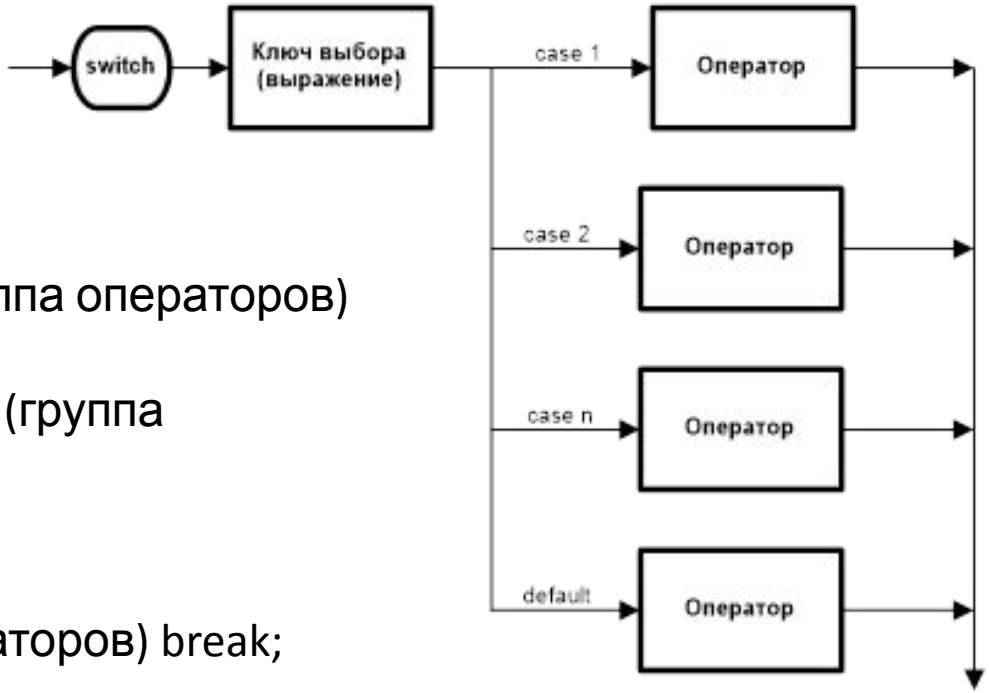
Не забывайте про фигурные скобки

Типы вычислительных процессов

Оператор выбора switch

позволяет выбрать одно из нескольких возможных выполнений программы

Параметром, по которому осуществляется выбор, служит так называемый **ключ выбора** (или селектор) - выражение любого простого типа




```
switch (выражение){
      case значение 1: оператор (группа операторов)
                                    break;
            case значение 2: оператор (группа
операторов)
                                    break;
            . . .
      default: оператор (группа операторов) break;
}
```

Типы вычислительных процессов

Циклические процессы

(часть операторов многократно выполняется при различных значениях переменных)

В языках программирования имеется три разновидности операторов цикла:

1. Оператор с предварительным условием (предусловием);
2. Оператор цикла с последующим условием (постусловием);
3. Оператор цикла с параметром.

Типы вычислительных процессов

Циклические процессы. Классификация типов циклических процессов

Могут быть вложены один в другой.

Необходимо составлять программу так, чтобы внутренний цикл полностью укладывался в циклическую часть внешнего цикла.

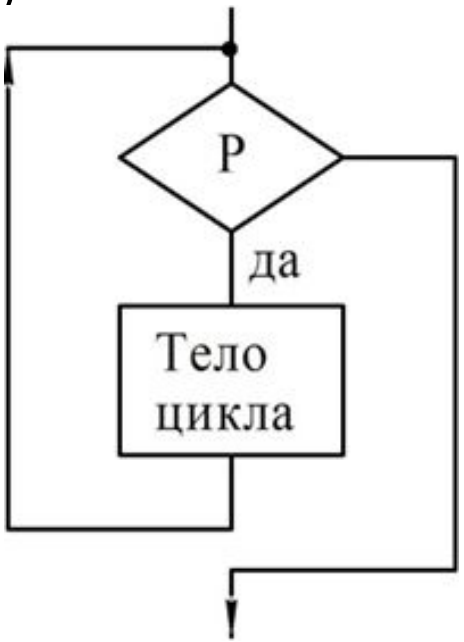
На количество вложений цикла в цикл в итерационных циклах ограничений нет

Типы вычислительных процессов

Циклические процессы

По способу **реализации** циклы делятся на:

Циклы с **предусловием** (тип «ПОКА»)



Циклы с **постусловием** (тип «ДО»)

Типы вычислительных процессов

Циклические процессы

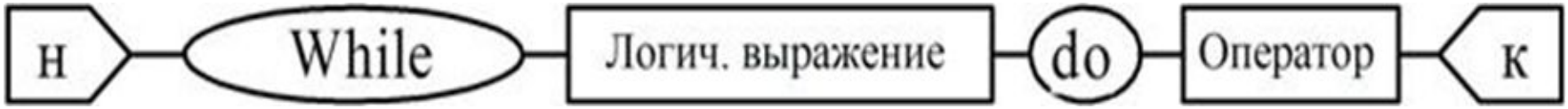
Циклы с **предусловием** (тип «ПОКА»).

Особенности

- Ни разу не выполняется, если условие ложно;

- Если ОПЕРАТОР составной, то он заключается в фигурные скобки. Зачастую оператор должен быть составным, поскольку он включает оператор, который должен повторяться и **оператор**, который **изменяет значение** в логическом выражении, **иначе** будет **зацикливание**;

- ОПЕРАТОР может быть циклом (причём на количество вложений цикла в цикл ограничений нет).



Типы вычислительных процессов

Циклические процессы

Циклы с **постусловием** (тип «ДО»).
Особенности

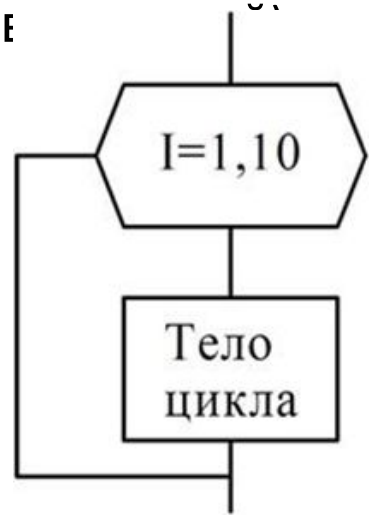
- Тело цикла исполняется хотя бы один раз;
- ОПЕРАТОР может быть циклом (причём на количество вложений цикла в цикл ограничений нет)

Типы вычислительных процессов

Циклические процессы

По виду **выполнения** циклы делятся на:

Арифметический (с параметром, с известным числом в



Итерационный (или с неизвестным числом вхождений).

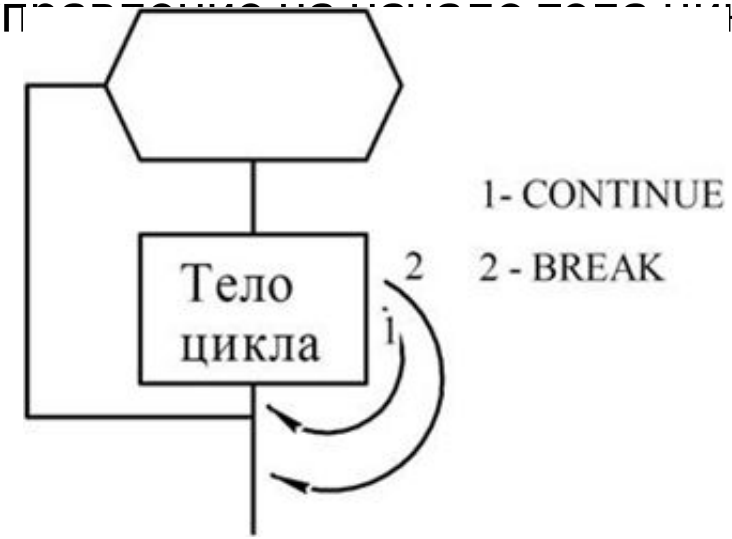
С помощью итерационных циклов решаются задачи, использующие метод последовательных приближений.

Типы вычислительных процессов

Циклические процессы. Арифметический (цикл с параметром). Особенности

Для досрочного выхода из цикла используются процедуры:

- **BREAK**: передаёт управление на оператор, следующий за циклом;
- **CONTINUE**: передаёт управление на начало тела цикла для следующей итерации



Типы вычислительных процессов

Логический тип данных

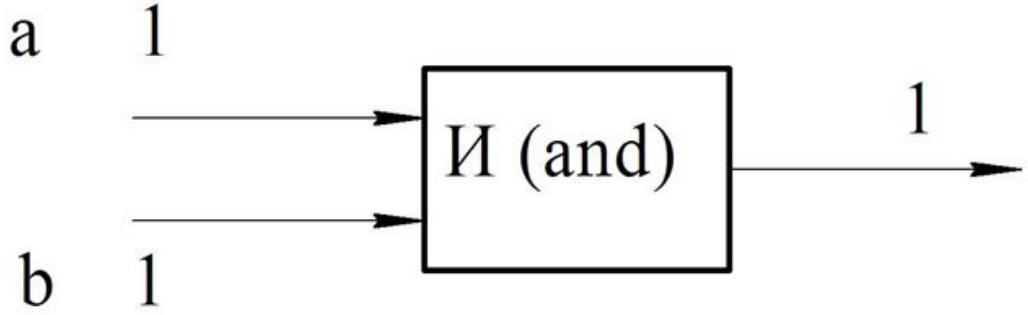
- Переменные логического типа описываются как bool
- Они могут принимать только два значения - False (ложь, 0) и True (истина, 1)
- Операции not, and, or и xor

Значения операндов		Результат операции			
X	Y	not X	X and Y	X or Y	X xor Y
False (0)	False (0)	True (1)	False (0)	False (0)	False (0)
False (0)	True (1)	True (1)	False (0)	True (1)	True (1)
True (1)	False (0)	False (0)	False (0)	True (1)	True (1)
True (1)	True (1)	False (0)	True (1)	True (1)	False (0)

Типы вычислительных процессов

Логический тип данных

Схема логического умножения **A and B**

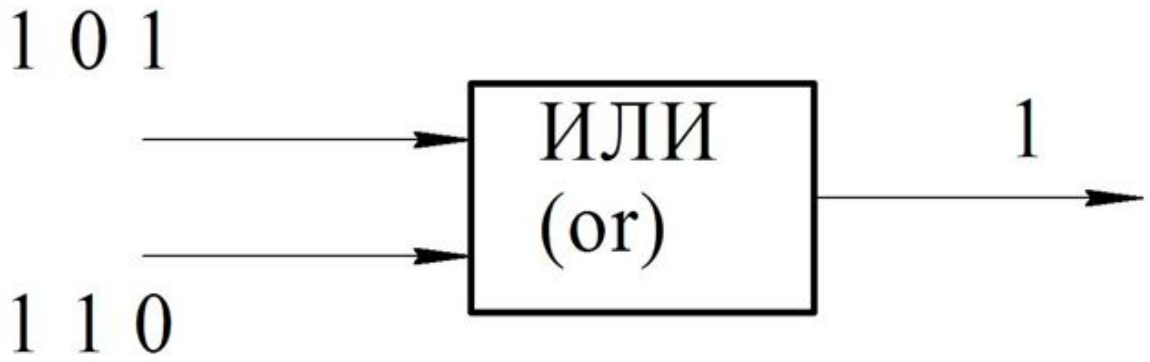


Результат операции and (и) есть **истина**, только если **оба** ее **операнда истинны**, и ложь во всех других случаях

Типы вычислительных процессов

Логический тип данных

Схема операции



Результат операции or (или) есть **истина**, если **какой-либо** из ее **операндов истинен**, и **ложен** только тогда, когда **оба операнда ложны**

Типы вычислительных процессов

Логический тип данных

Логические операции имеют более высокий приоритет, поэтому отношения, стоящие слева и справа от знака логической операции, должны быть заключены в скобки

Принят следующий приоритет операций:

- not
- and, *, /, mod
- or, xor, +, -
- Операции отношения

and - логические умножение

or - логическое сложение

xor - логическое сложение по mod 2

Типы вычислительных процессов

Логический тип данных

A or B and not (A or B)

4 3 2 1

NOT ((X>2) AND (X<4)) → (X<=2) OR (X>=4)

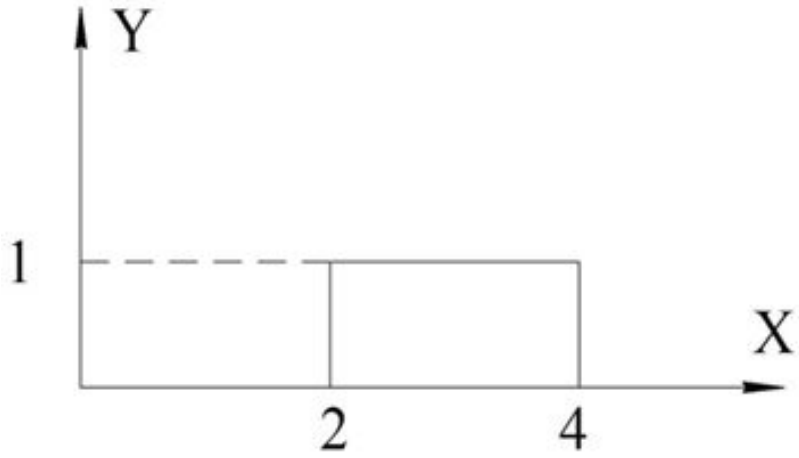
- При отрицании сложных предикатов операции отношения меняются на противоположные, а логические OR на AND
- Если один условный оператор стоит в ветке «да» второго оператора, то мы можем поместить все в один предикат, в котором условия объединяться логической операций AND
- Если один условный оператор стоит в ветке «нет» второго оператора, то они могут быть объединены с помощью операции OR

Типы вычислительных процессов

Логический тип данных

Рассмотрим задачу

Для функции единичного скачка вывести y для x, лежащего в следующей области: от 2 до 4, не включая концы отрезков



Типы вычислительных процессов

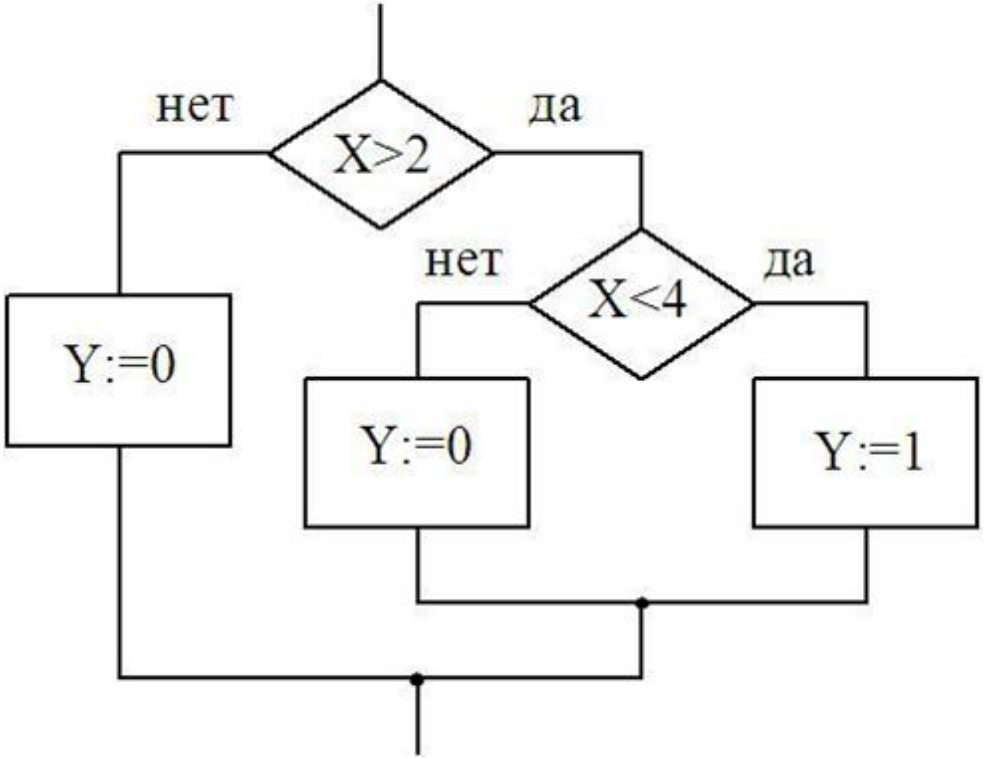
Логический тип данных

После выполнения выше сказанного, предикат будет выглядеть следующим образом:

(x>2) and (x<4)

или

(x <= 2) or (x >= 4)



Типы вычислительных процессов

Рекуррентные последовательности, рекуррентные алгоритмы

Рекуррентная последовательность

- Пусть известно k чисел a_1, ..., a_k
- Эти числа являются первыми числами числовой последовательности
- Следующие элементы данной последовательности вычисляются так:

$$a_{k+1} = F(a_1, \ldots, a_k);\ a_{k+2} = F(a_2, \ldots, a_{k+1});\ a_{k+3} = F(a_3, \ldots, a_{k+2}); \ldots$$

F — функция от k аргументов

- Формула вида $a_i = F(a_{i-1}, a_{i-2} \ldots, a_{i-k})$ называется **рекуррентной формулой**
- Величина k называется **глубиной рекурсии**

Типы вычислительных процессов

Рекуррентные последовательности, рекуррентные алгоритмы

Рекуррентная последовательность — это бесконечный ряд чисел, каждое из которых, за исключением k начальных, выражается через предыдущие

Примерами рекуррентных последовательностей являются **арифметическая**(1) и **геометрическая** (2) прогрессии

$a_1 = 1, a_2 = 3, a_3 = 5, a_4 = 7, a_5 = 9, \ldots$ (1)

Рекуррентная формула для указанной арифметической прогрессии: $a_i = a_{i-1} + 2$

$a_1 = 1, a_2 = 2, a_3 = 4, a_4 = 8, a_5 = 16, \ldots$ (2)

Рекуррентная формула для данной геометрической прогрессии: $a_i = a_{i-1} * 2$

Типы вычислительных процессов

Рекуррентные последовательности, рекуррентные алгоритмы

Рекуррентная последовательность

- В целом рекуррентная последовательность описывается совокупностью начальных значений и рекуррентной формулы
- Для арифметической и геометрической прогрессии формулы можно объединить в ветвящуюся формулу

Для арифметической прогрессии

$$a_i = \begin{cases} 1, \text{если } i = 1 \\ a_{i-1} + 2, \text{если } i > 1 \end{cases}$$

Для геометрической прогрессии

$$a_i = \begin{cases} 1, \text{если } i = 1 \\ a_{i-1} * 2, \text{если } i > 1 \end{cases}$$

Глубина рекурсии в обоих случаях равна единице (такую зависимость еще называют **одношаговой рекурсией**)

Типы вычислительных процессов

Рекуррентные последовательности, рекуррентные алгоритмы

Рекуррентная последовательность

Числа Фибоначчи

Начиная с третьего элемента каждое число равно сумме значений двух предыдущих

$$a_i = \begin{cases} 1, \text{если } i = 1, i = 2 \\ f_{i-1} + f_{i-2}, \text{если } i > 2 \end{cases}$$

- Это рекуррентная последовательность с глубиной равной 2 (двухшаговая рекурсия)

Типы вычислительных процессов

Рекуррентные последовательности, рекуррентные алгоритмы

Рекуррентная последовательность

Факториал целого числа n!

Это значение n-го элемента следующего ряда чисел

$$a_0 = 1, a_1 = 1!, a_2 = 2!, a_3 = 3!, a_4 = 4!, \ldots$$

Тогда рекуррентное описание такой последовательности

$$a_i = \begin{cases} 1, \text{если } i = 0 \\ a_{i-1} * i, \text{если } i > 0 \end{cases}$$

Типы вычислительных процессов

Рекуррентные последовательности, рекуррентные алгоритмы

С рекуррентными последовательностями связаны задачи такого рода:

1) вычислить заданный (n-й) элемент последовательности;

2) математически обработать определенную часть последовательности (например, вычислить сумму или произведение первых n членов);

3) подсчитать количество элементов на заданном отрезке последовательности, удовлетворяющих определенным свойствам;

4) определить номер первого элемента, удовлетворяющего определенному условию;

5) вычислить и сохранить в памяти заданное количество элементов последовательности

Типы вычислительных процессов

Рекуррентные последовательности, рекуррентные алгоритмы

Приме
р

Для заданного натурального N и вещественного x (x > 0) вычислить значение выражения

$$\sqrt{x+\sqrt{x+\ldots+\sqrt{x}}}.$$

$$a_1 = \sqrt{x},\ a_2 = \sqrt{x+\sqrt{x}},\ a_3 = \sqrt{x+\sqrt{x+\sqrt{x}}},\ldots$$

$$a_2 = \sqrt{x+a_1},\ a_3 = \sqrt{x+a_2},\ \ldots,\ a_i = \sqrt{x+a_{i-1}}.$$

Типы вычислительных процессов

Методы проверки правильности алгоритмов

Корректность программы – это соответствие ее функциональности требованиям

Это и проверка исходного кода, и поиск типовых ошибок в коде, и тестирование программы (имеется в виду запуск программы на каких-либо примерах), и имитационное моделирование, и формальная верификация (логическая модель программы и анализ ее математическими средствами)

Типы вычислительных процессов

Методы проверки правильности алгоритмов

Аналитическое доказательство правильности работы программы называют **верификацией**

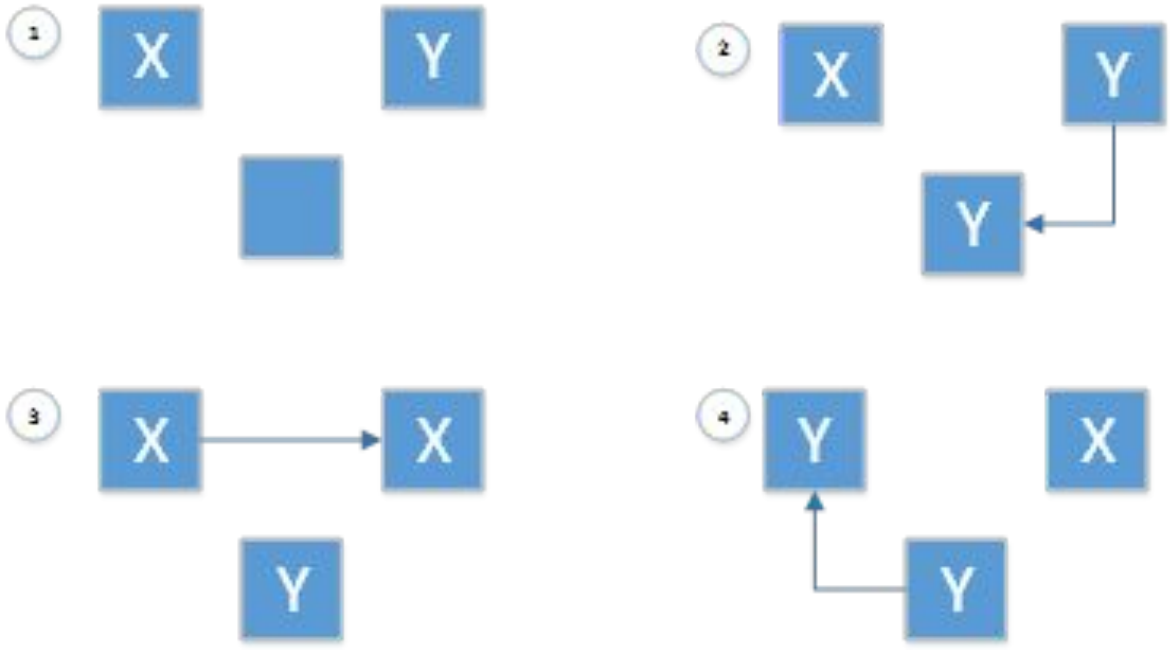
a) Доказательство правильности работы программ методом логического преобразования предикатов. Основано на логике высказываний.

b) Верификация с помощью **таблицы трассировки**

Типы вычислительных процессов

Методы проверки правильности алгоритмов

Пример 1. Надо поменять
содержимое ячеек x и y местами



Типы вычислительных процессов

Методы проверки правильности алгоритмов

Пример 2. Докажем правильность менее простого алгоритма

[0] x = x + y
[1] y = x – y
[2] x = x – y

Покажем, что $x_3 = y_0$, $y_3 = x_0$

Предписание	x	y
x = x + y	$x_1 = x_0 + y_0$	$y_1 = y_0$
y = x – y	$x_2 = x_1$	$y_2 = x_1 - y_1$
x = x – y	$x_3 = x_2 - y_2$	$y_3 = y_2$

$x_3 = x_2 - y_2 = x_1 - (x_1 - y_1) = y_1 = y_0$, т.е. $x_3 = y_0$

$y_3 = x_1 - y_1 = x_0 + y_0 - y_0 = x_0$, т.е. $y_3 = x_0$

Типы вычислительных процессов

Методы проверки правильности алгоритмов

Тестирование вручную

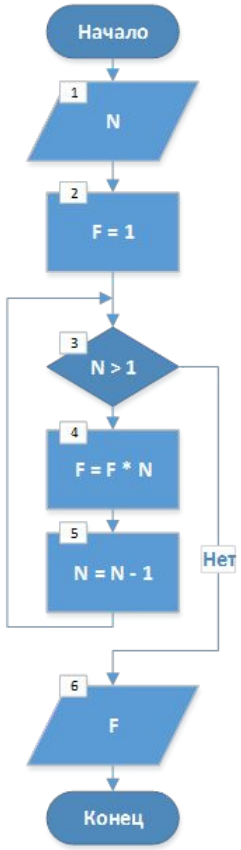
- Производится расчет значений выходных данных по фиксированному набору входных данных
- В случае выявления ошибки производится ее поиск с помощью промежуточных данных (метод локализации ошибки)
- Для такого поиска используют **таблицу трассировки**, которая позволяет проследить изменения переменной в процессе выполнения программы

Типы вычислительных процессов

Методы проверки правильности алгоритмов

Пример. Вычислим факториал числа N

```
Ввод N
F = 1
Пока N > 1
    F = F * N
    N = N – 1
Вывод F
```



Пример трассировки программы для значения N равного 7

7! = 7*6*5*4*3*2*1 = 5040

Типы вычислительных процессов

Методы проверки правильности алгоритмов

Предписание	N	F
Ввод N	7	
F = 1	7	1
ПОКА N > 1	7 > 1, цикл выполняется	1
F = F * N	7	1 * 7 = 7
N = N - 1	7 – 1 = 6	7
ПОКА N > 1	6 > 1, цикл выполняется	7
F = F * N	6	7 * 6 = 42
N = N - 1	6 – 1 = 5	42
ПОКА N > 1	5 > 1, цикл выполняется	42
F = F * N	5	42 * 5 = 210
N = N - 1	5 – 1 = 4	210
…	…	…
N = N - 1	2 – 1 = 1	5040
ПОКА N > 1	1 = 1, цикл не выполняется	5040
Вывод F	1	5040

Типы вычислительных процессов

Методы проверки правильности алгоритмов

Третий шаг отладки программы — её тестирование

- Необходимо продумать и подготовить набор тестов — такой набор возможных исходных данных, который исчерпывает все возможные ситуации
  - необходимо тщательно продумать и подготовить набор тестов — такой набор возможных исходных данных, который исчерпывает все возможные ситуации
  - граничные значения данных (если заранее известны диапазоны их изменения)
  - значения данных вне границ допустимого диапазона

Если программа предполагает ввод нескольких исходных данных, то необходимо предусмотреть в наборе тестов все возможные их комбинации