

# Тема 1-2.

## Этапы разработки программы Структурное программирование

для АСУБ и ЭВМб

# В предыдущих лекциях: Что такое программирование?

## Процесс создания программы для ЭВМ:

– Программа = Данные + Алгоритм + Язык

– ЭВМ

- Архитектура: процессор + память + остальное

- фон Неймана (Принстонская)

- Гарвардская

– Язык

- Низкого уровня: машинный, ассемблер

- Высокого уровня: C++



# В предыдущих лекциях: принцип разработки программ

- Разработка программы на языке высокого уровня никогда не должна начинаться собственно с программирования
- **Структурное программирование** – методология программирования, базирующаяся на системном подходе к анализу, проектированию и реализации программного обеспечения.
- Структурное программирование предполагает использование **наиболее простых структур** как при проектировании, так и при программировании и отладке программ

# Этапы разработки программы

## 1. Внешнее проектирование

- Определение общей структуры и взаимодействия функций

## 2. Составление внешних спецификаций

- Таблица спецификаций

## 3. Внутреннее проектирование

- Разработка алгоритма решения задачи

## 4. Кодирование

## 5. Отладка

## 6. Тестирование

# Внешнее проектирование

- На этом этапе рассматривается возможность разбиения задачи на **подзадачи**, каждая из которых решает некоторую законченную часть всей задачи и определяется **иерархия** этих подзадач (какая подзадача вызывает для выполнения другие подзадачи). Одна из подзадач всегда является основной, **главной**.
- Кроме того, на этом этапе выясняется, какие **данные** нужно передать вызываемой подзадаче и какие данные получить от нее. Для каждой подзадачи описывается своя **таблица внешних спецификаций**. Если задача простая, то этот этап можно опустить.

# Составление внешних спецификаций

– приведение условия задачи к некоторому стандарту, что позволяет задачи из различных предметных областей описывать более или менее унифицировано.

Таким стандартом для простых программ или отдельных функций являются *внешние спецификации*.

*Спецификации* определяют входные и выходные данные задачи и оформляются в виде таблицы:

# Таблица спецификаций

№	Имя	Назначение	Тип	Вх/Вых.	Диапазон
---	-----	------------	-----	---------	----------

# Таблица спецификаций

- *Входные величины* – это те, значения которых необходимо задать в начале решения задачи.
- *Выходные величины* – это те, которые являются результатом решения задачи.
- *Имя* ставится в соответствие любой входной и выходной величине; имена сохраняются на всех этапах решения задачи.
- *Назначение* – словесное описание величины, для чего она нужна.
- *Тип* – определяет тип данных (целое число, действительное число, текст)
- *Диапазон* определяет область изменения величины (целое число, действительное число, текст, целое число от 0 до 5 и т.д.)



# Разработка алгоритма решения задачи

- *Алгоритм* – это описание последовательности действий, необходимых для решения задачи
- Алгоритм обладает свойствами
- Аль-Хорезми

# Реализация алгоритма

- **Программирование алгоритма.** На этом этапе (называемом также кодированием), происходит запись алгоритма на каком-либо языке программирования.
- **Синтаксическая отладка программы,** то есть исправление ошибок, связанных с неверным использованием конструкций языка программирования. Эти ошибки выявляются на этапе компиляции программы.

# Реализация алгоритма

- **Тестирование программы**, то есть проверка ее работоспособности на различных вариантах исходных данных с просчитанными вручную результатами
  - Эти результаты должны совпасть с результатами, полученными при расчете на компьютере
  - При их несовпадении следует искать в программе логические ошибки, связанные с неверным алгоритмом
- Количество тестов должно быть таким, чтобы каждый тест проверял какую-то специфическую ситуацию, и не было ситуаций, не проверенных ни одним из тестов.

# Таблица тестов

Номер теста	Назначение теста	Входные данные	Выходные данные

# Программирование

## Процесс создания программы для ЭВМ:

- Программа = Данные + **Алгоритм**
- ЭВМ
  - Архитектура: процессор + память + остальное
    - фон Неймана (Принстонская)
    - Гарвардская
- Язык
  - Низкого уровня: машинный, ассемблер
  - Высокого уровня: C++



# Разработка алгоритма решения задачи

- *Алгоритм* – это описание последовательности действий, необходимых для решения задачи
- Алгоритм обладает свойствами
- Аль-Хорезми

# Свойства алгоритма

- **Массовость** – алгоритм должен описывать не одну конкретную задачу, а группу подобных задач
- Например, не решение одного конкретного квадратного уравнения,

$$2X^2 - 4X + 3 = 0$$

- а любого квадратного уравнения

$$AX^2 + BX + C = 0$$

# Свойства алгоритма

- **Детерминированность** - алгоритм должен всегда давать один и тот же результат при одних исходных данных
- **Результативность** - алгоритм должен давать какой-то результат для любого варианта исходных данных.
  - Например, если решается квадратное уравнение, то должны быть предусмотрены случаи:



# Свойства алгоритма: пример

- Существует два различных действительных корня

$$B^2 - 4AC > 0$$

- Существует два равных действительных

$$B^2 - 4AC = 0$$

- Нет действительных корней

$$B^2 - 4AC < 0$$

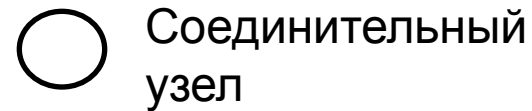
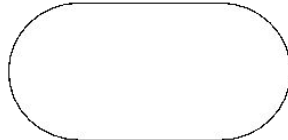
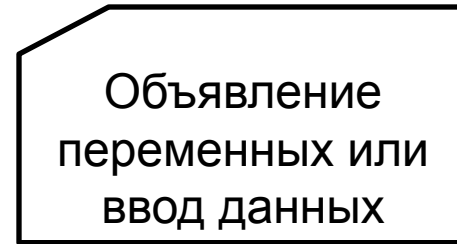
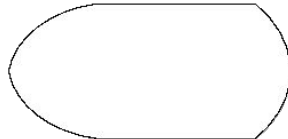
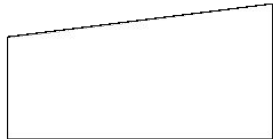
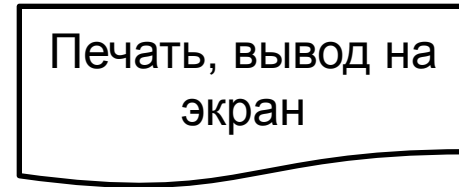
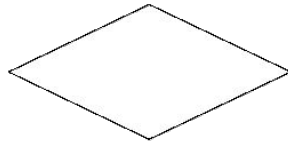
- Уравнение вырождено

$$A = 0$$

# Описание алгоритма

- **Словесное описание алгоритма** - это последовательность пронумерованных шагов, описывающих решение задачи.
  - Шаги могут иметь многоуровневую нумерацию, чтобы показать, что одни шаги являются частью других шагов
- **Блок-схема решения задачи** – графическое изображение алгоритма в виде взаимосвязанных блоков.

# Элементы блок-схемы



# Утверждение структурного программирования

## Теорема Бёма — Якопини

Алгоритм любой сложности можно реализовать, используя только три конструкции:

- следования (линейные)
- выбора (ветвления)
- повторения (циклические)

**Линейный** - алгоритм, в котором все указанные действия выполняются один раз в том порядке, в котором они записаны.

# Виды вычислительных процессов

- **Линейный процесс** - последовательное размещение шагов
- **Разветвляющийся процесс** - в зависимости от условия нужно выполнять либо одно, либо другое действие
- **Циклический процесс** - это такой процесс, в котором некоторая последовательность действий выполняется несколько раз до тех пор, пока выполняются некоторые условия

Все задачи являются комбинацией этих трех видов процессов

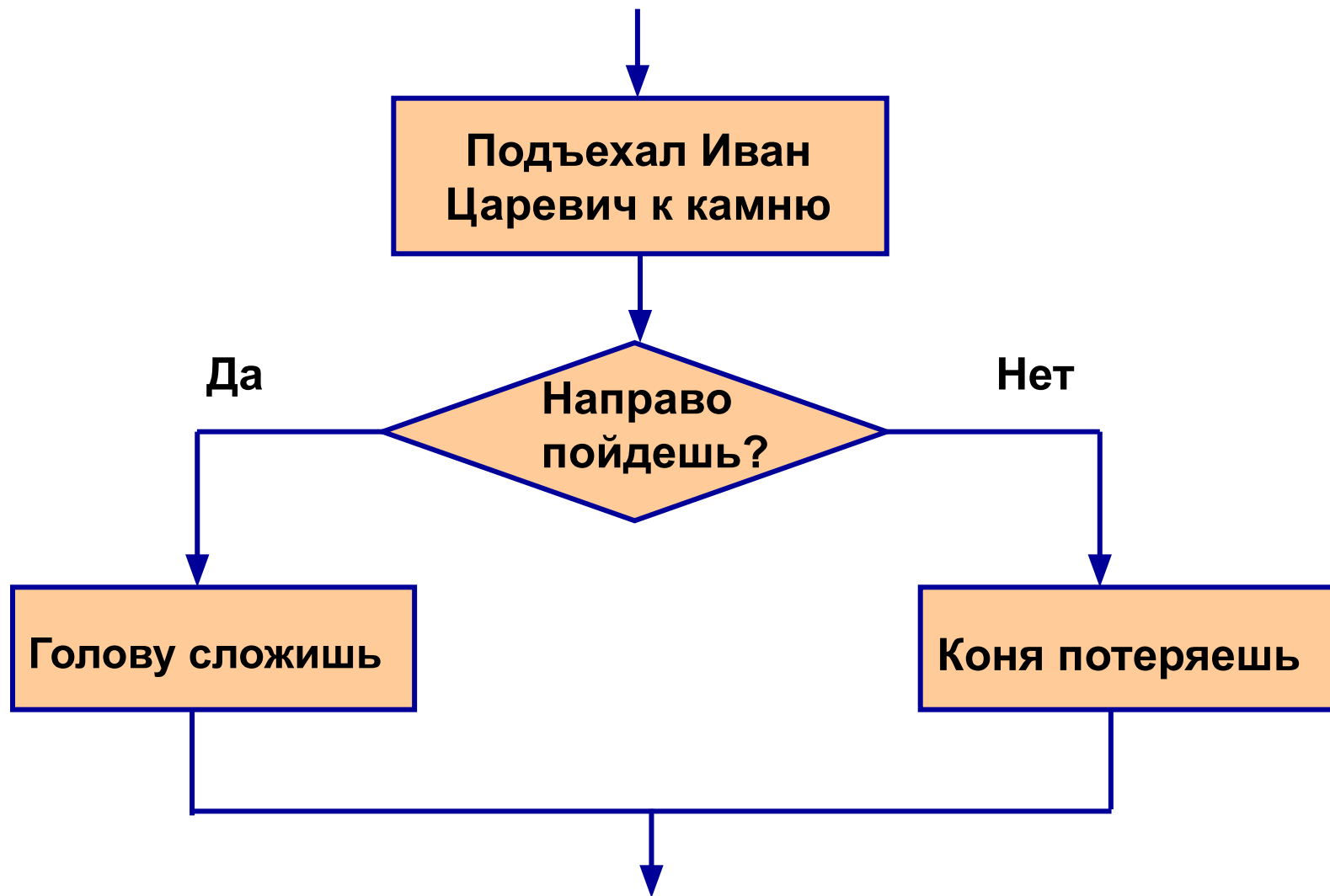
## Например, алгоритм посадки дерева:

- 1) Выкопать в земле ямку;
- 2) Опустить в ямку саженец;
- 3) Засыпать ямку с саженцем землей;
- 4) Полить саженец водой.



***Разветвляющийся процесс*** - в зависимости от условия нужно выполнять либо одно, либо другое действие







# Разветвляющийся процесс

ЕСЛИ *условие* ТО

*Действия*

ИНАЧЕ

*Действия*

ЕСЛИ ВСЕ

В качестве действий может стоять проверка другого условия

# Разветвляющийся процесс

ВЫБОР ПО переменная

Значение\_1: Действия

Значение\_2: Действия

Значение\_n: Действия

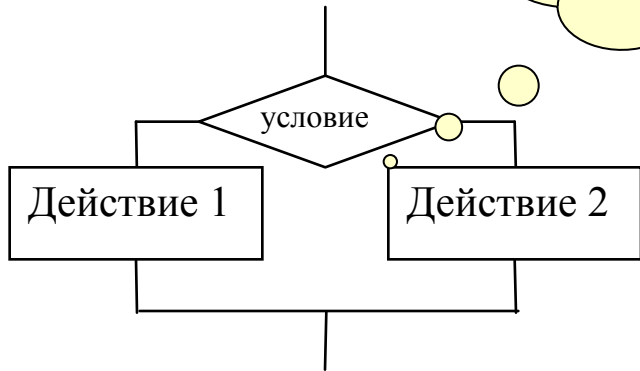
\*: Действия

ВЫБОР ВСЕ

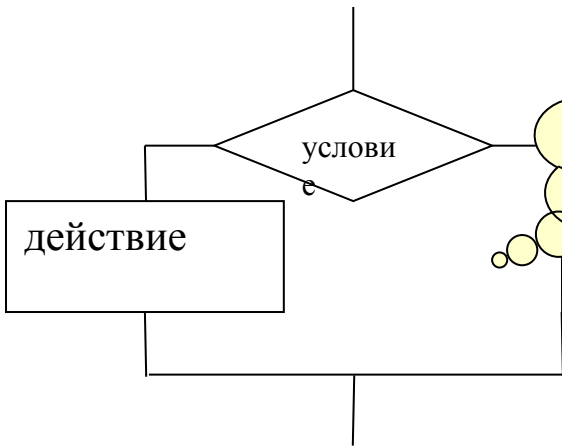
\* означает, что переменная не равна ни одному значению

Чтобы избежать вложенных условия, используется конструкция ВЫБОР. Она позволяет иметь несколько ветвей для проверки равенства переменной одному из многих значений.

# Ветвление

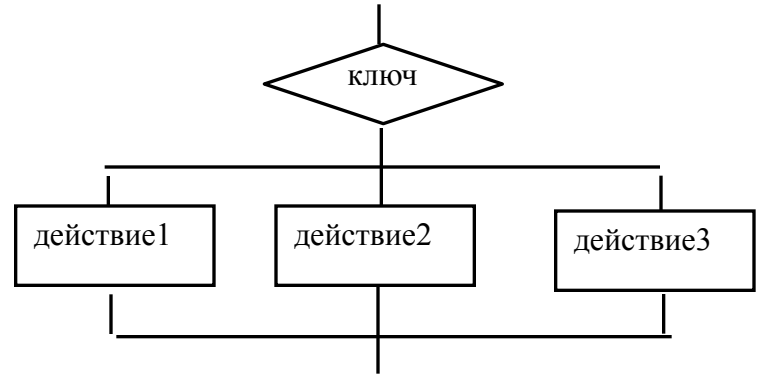


Полная форма

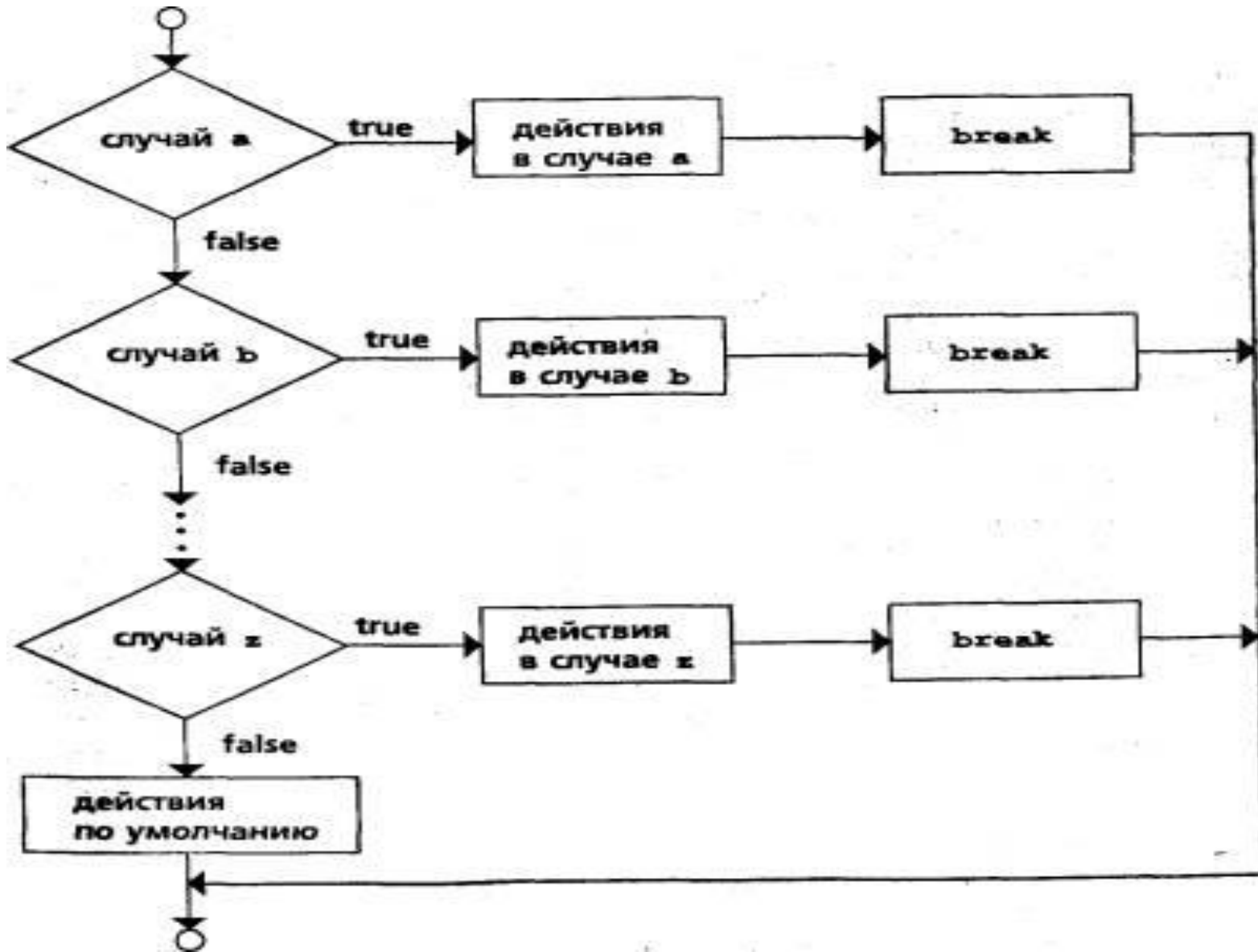


Неполная форма

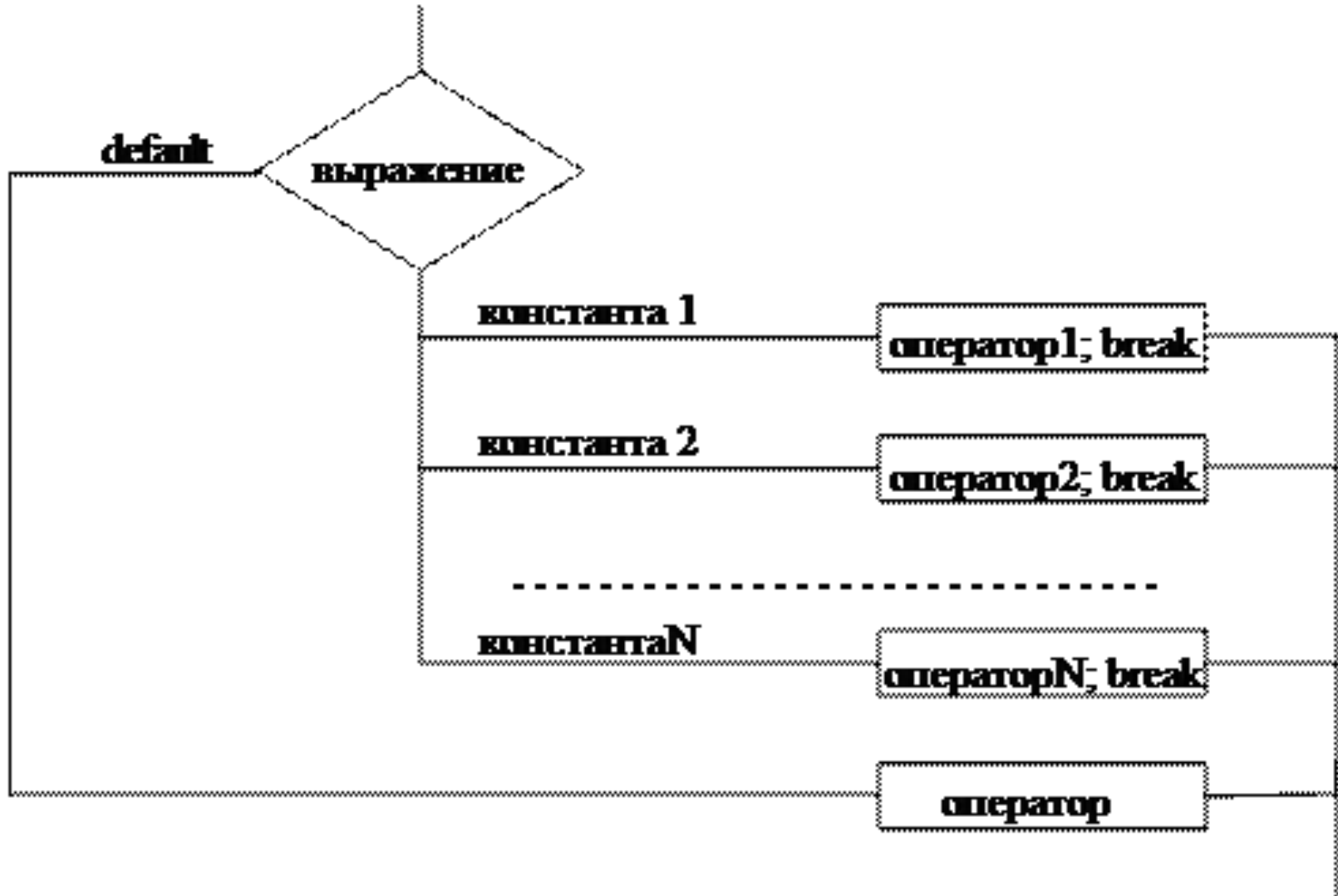
# Выбор



# Выбор (вариант блок схемы)



# Выбор (вариант блок схемы)



**Циклический** процесс – это такой процесс, в котором некоторая последовательность действий может выполняться несколько раз в зависимости от заданного условия.



**ПОКА** условие **ВЫПОЛНИТЬ**  
Действия  
**ПОКА** **ВСЕ**

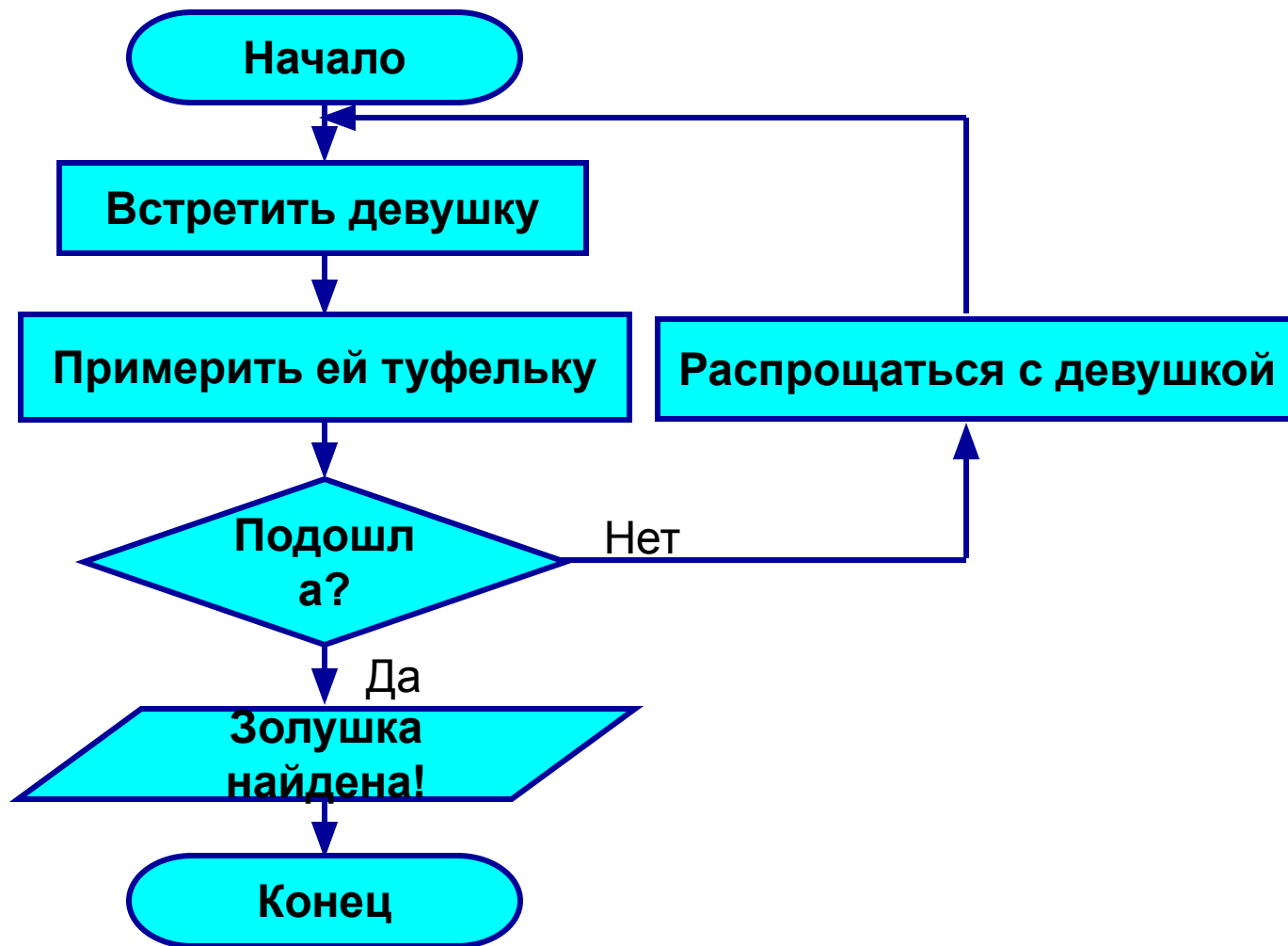
## Цикл "Пока" (цикл с предусловием)



## Цикл с постусловием

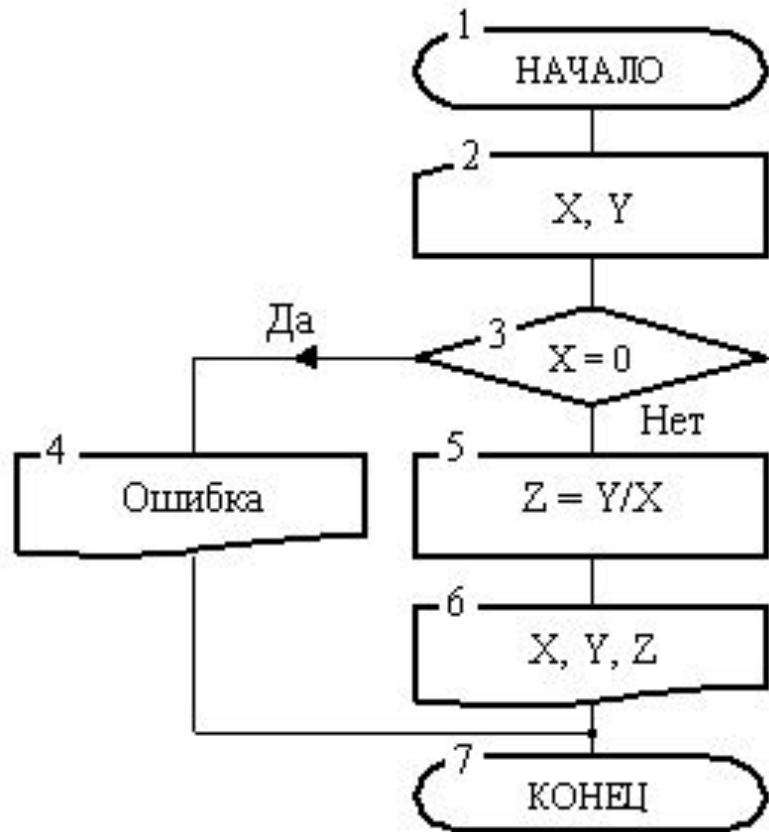
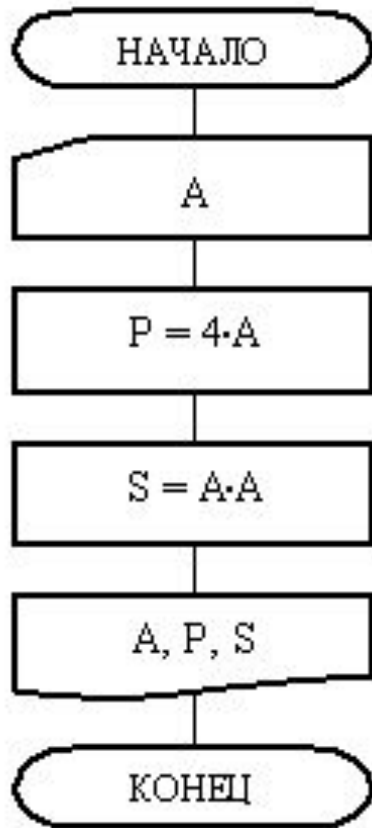


# Алгоритм поиска Золушки:

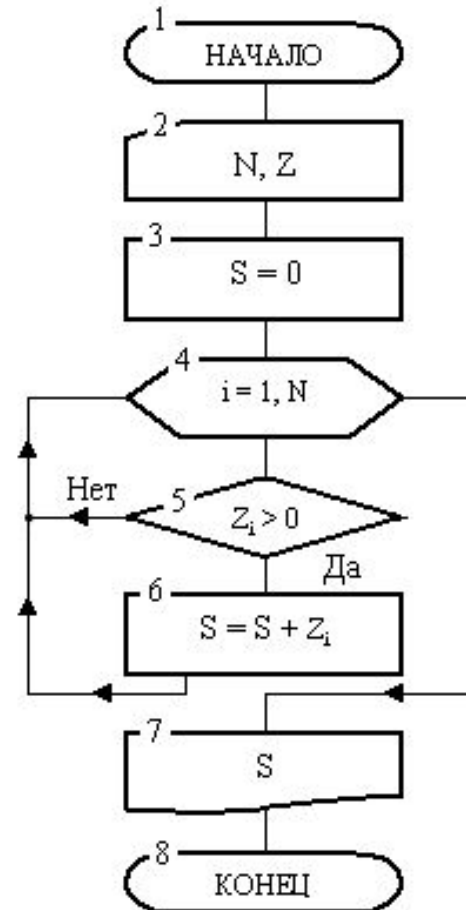
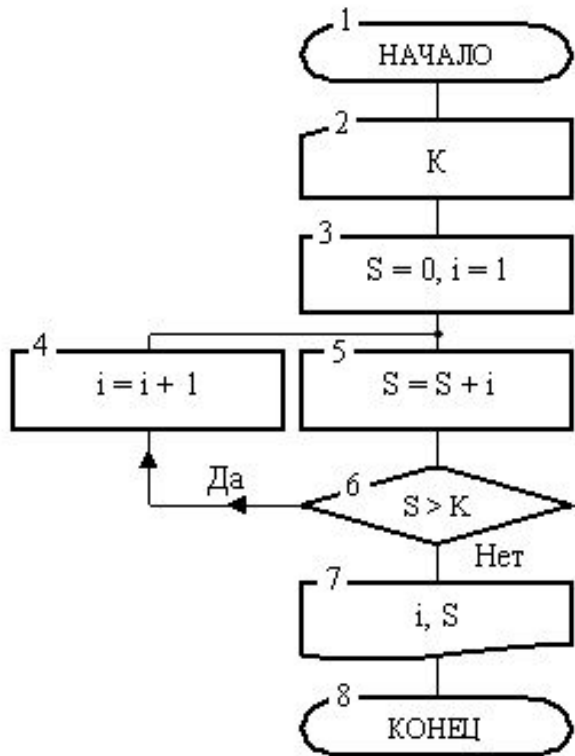




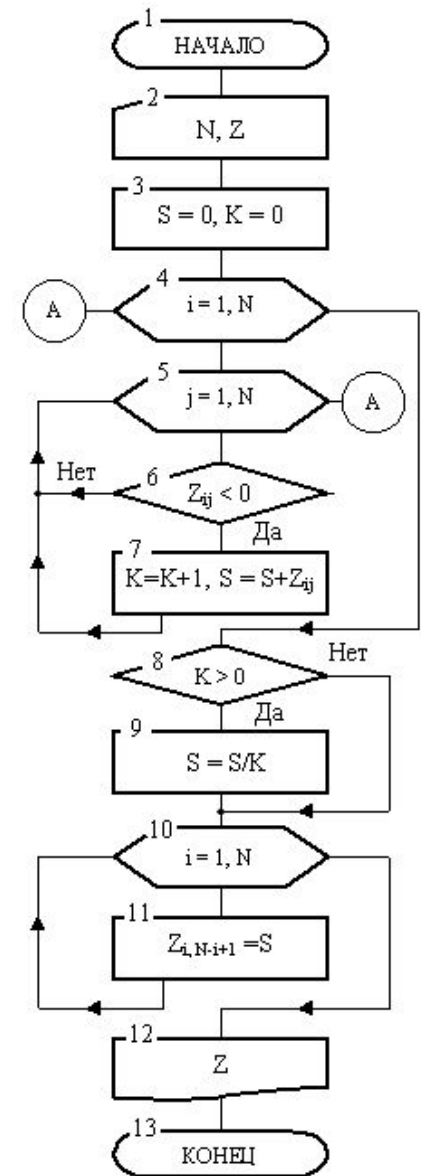
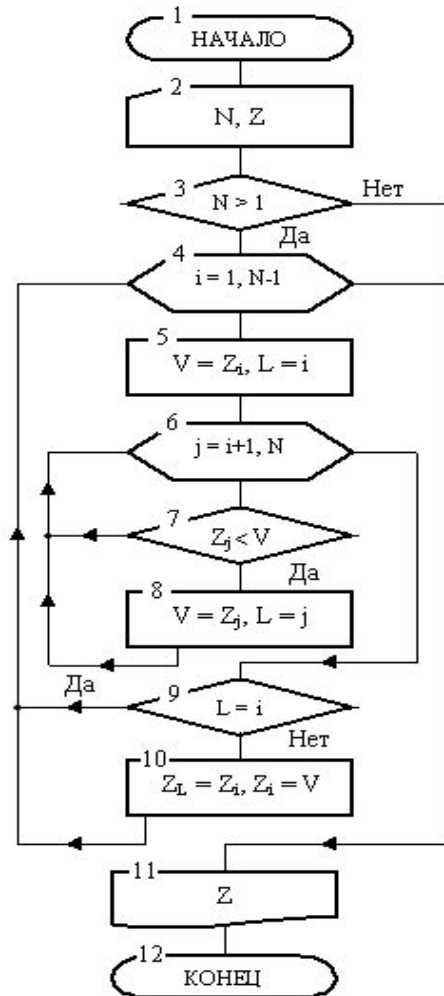
# Линейный и разветвляющийся алгоритмы



# Циклические алгоритмы



# Алгоритмы со структурами ВЛОЖЕННЫХ ЦИКЛОВ



# Разветвляющийся процесс: пример

- Студенты Иванов и Петров за время практики заработали определенную сумму.
- Кто из них заработал большую сумму?
- Определить средний заработок.

# Разветвляющийся процесс: таблица спецификаций

№	Имя	Назначение	Тип	Вх/Вых	Диапазон
1	X	Сумма, заработанная Ивановым	Действ. число	вход	>0
2	Y	Сумма, заработанная Петровым	Действ. число	вход	>0
3	M	Средний заработок	Действ. число	выход	>0
4	S	Сообщение о соотношении заработка	Текст	выход	{S.1, S.2, S.3}

# Разветвляющийся процесс: словесное описание алгоритма

1. Ввод X и Y.
2.  $M = (X + Y) / 2$
3. ЕСЛИ  $X > Y$  ТО
  - 3.1.  $S = \text{«Иванов заработал больше»}$ .
  - 3.2. Переход к п. 5
- ЕСЛИ ВСЕ
4. ЕСЛИ  $Y > X$  ТО
  - 4.1.  $S = \text{«Петров заработал больше»}$
  - 4.2. Переход к п. 5
- ИНАЧЕ
- 4.3.  $S = \text{«Они заработали поровну»}$
- ЕСЛИ ВСЕ
5. Вывод S и M

# Разветвляющийся процесс: тесты

Номер теста	Назначение теста	Входные данные	Выходные данные
1	Зарботок Иванова больше, чем зарботок Петрова	$X=200$ $Y=100$	$M=150$ $S=$ Иванов зарботал больше
2	Зарботок Петрова больше, чем зарботок Иванова	$X=100$ $Y=200$	$M=150$ $S=$ Петров зарботал больше
3	Зарботки равны	$X=200$ $Y=200$	$M=200$ $S=$ Они зарботали поровну
4	Корректность ввода	$X=-200$ $Y=200$	«Зарботок должен быть больше нуля»

# Реализация задачи на c++

```
#include <iostream>
#include <string>
int main()
{
    float X, Y, M;
    string S;
    std::cin >> X ;
    std::cin >> Y ;
    M = (X + Y) / 2;
    if (X > Y) S = "Иванов заработал больше";
    else
        if (X < Y) S = "Петров заработал больше";
        else S = "Они заработали поровну";

    std::cout<<"Результат: "<<S<<std::endl;
    std::cout<<"Среднее: "<< M <<std::endl;
    return 0;
}
```