Последовательность разработки программ

Лесли Анна Робертсон в своей книге "Программирование – это просто. Пошаговый подход" предлагает выделять

7 основных этапов разработки программ:

- **Постановка задачи,**
- **План решения задачи,**
- **Разработка алгоритма по плану,**
- **Проверка правильности алгоритма,**
- **Перевод алгоритма на конкретный язык программирования,**
- **Запуск, отладка и тестирование программы на компьютере,**
- **Создание документации для программы.**

Последовательность разработки программ

Рассмотрим подробнее эти 7 этапов:

1. Постановка задачи

Определение целей решения задачи, функций программы и обобщённой структуры алгоритма решения задачи.

Многократное внимательное изучение и анализ текста задачи "до тех пор, пока полностью не будет осмыслено то, что требуется сделать".

При анализе задачи необходимо выделить :

- **действия по вводу данных и характеристики входных данных (количество, вид, назначение и др.),**
- **действия по выводу данных и характеристики выходных данных (количество, вид, назначение и др.),**
- **действия по обработке данных для получения выходных данных их входных.**

На этом этапе анализа составляется обобщённая функциональная схема алгоритма решения задачи: задача структурируется (разбивается на относительно независимые (слабо зависимые) части и описываются функции всей задачи и каждой обобщённой части задачи, а также действия по реализации каждой функции).

В том случае, когда задача не описана в текстовом виде или нечётко сформулирована, надо в режиме диалога-интервью с заказчиком составить чёткое текстуальное (вербальное) описание задачи и согласовать его с заказчиком программы.

Последовательность разработки программ

2. План решения задачи

Детализация обобщённой функциональной структуры, определение этапов обработки.

Первоначальный план решения задачи представляет собой грубый набросок, который затем детализируется, уточняется, перерабатывается.

Обычно план решения включает в себя:

- **Основную логику решения задачи,**
- **Основные этапы обработки данных,**
- **Основные подзадачи (если требуется разбиение на подзадачи),**
- **Пользовательский интерфейс (если есть),**
- **Основные переменные и структуры данных.**

Последовательность разработки программ

3. Разработка алгоритма по плану

План решения развивается в алгоритм: совокупность отдельных действий и операций, которые полностью описывают решаемые задачи и порядок их решения (выполнения).

Для того, что бы написать алгоритм и программу для ЭВМ надо:

- описать решаемую задачу в терминологии деятельности – определить основные действия (группы действий) по решению задачи,
- разбить задачу и подзадачи на совокупность действий,
- каждое действие разбить на отдельные операции,
- определить алгоритмы выполнения каждой операции,
- детализация действий: описание основных действий в действиях более низкого уровня,– выполняется до тех пор, пока каждый из шагов алгоритма не описывается как детальная операция: способ выполнения действия с учётом внешних условий (ситуации),
- определить последовательность/параллельность выполнения действий и операций либо механизмы (принципы и протоколы) взаимодействия отдельных подзадач в процессе решения задачи.

Последовательность разработки программ

4. Проверка правильности алгоритма

Цель проверки алгоритма "на бумаге" – заблаговременный поиск основных логических ошибок, пока их нетрудно обнаружить и исправить.

Проверка правильно ли выполняет алгоритм задуманные действия и операции осуществляется следующим образом:

- **Подготавливаются тестовые входные данные,**
- **"Вручную" рассчитываются выходные данные соответствующие тестовым входным данным,**
- **Программист на бумаге ("вручную") подставляет входные данные в алгоритм и проверяет изменение всех переменных алгоритма, имитируя работу программы и проверяя логику работы алгоритма,**
- **Выходные данные, полученные в результате имитации работ алгоритма, сравниваются с выходными данными рассчитанными вручную. Таким образом, обнаруживаются логические ошибки в работе алгоритма.**

Для проверки алгоритма "на бумаге" обычно используются проверочные таблицы, отображающие состояние переменных алгоритма (программы) на каждом шаге выполнения.

5. Перевод алгоритма на конкретный язык программирования

Производится кодирование алгоритма на выбранном языке программирования с учётом особенностей этого языка.

Последовательность разработки программ

6. Запуск, отладка и тестирование программы на компьютере

Суть отладки состоит в выполнении программы на компьютере и устранении синтаксических ошибок ЯП и логических ошибок алгоритма.

Отладка – это поиск и исправление синтаксических (проявляются во время компиляции) и логических (проявляются во время выполнения) ошибок в программе.

Тестирование – это проверка работы программы со специально составленными наборами входных данных (тестов) и сравнение полученных результатов с рассчитанными "вручную" в поисках возможных логических или семантических ошибок.

Этот этап выполняется несколько раз пока не останется сомнений в правильности работы программы.

Последовательность разработки программ

7. Создание документации для программы

Документация программы должна включать, *как минимум*:

- **комментарии в текстах всех программных модулей (комментарий-заголовок, комментарии ко всем типам данных, переменным и константам программы, комментарии к ключевым блокам кода программы);**
- **инструкцию по установке программы,**
- **описание программы для системных администраторов,**
- **руководство пользователя.**

Документирование – завершающий этап создания программы, но разработка документации начинается с началом постановки задачи.

В процессе эксплуатации программы документация обычно дополняется описанием и изменений, вносимых в программу после её сдачи заказчику.

Решение задач с логически разными входными данными

Входные данные при решении задач могут не только содержать все необходимые для решения значения, но излишние или недостающие данные.

Логические типы варьирования входных данных *представлены четырьмя возможными случаями:*

1. **Включение в условия всех необходимых признаков (данных) для ответа на вопрос задачи,**
2. **Включение в условия не всех необходимых признаков,**
3. **Включение в условия всех необходимых и избыточных признаков,**
4. **Включение в условия не всех необходимых и избыточных признаков.**

Во втором случае необходимо использовать один из следующих алгоритмов:

- **получить недостающие данные путём вычислений из имеющихся,**
- **на основе имеющейся информации принять гипотезу(ы) о составе и значениях недостающих данных,**
- **декларировать невозможность решения задачи, поскольку ни вычислить недостающие данные, ни сформировать достоверную гипотезу невозможно.**

В третьем случае необходимо отсортировать избыточные данные от необходимых и использовать последние для решения задачи.

В четвёртом случае необходимо отсортировав избыточные данные использовать алгоритмы описанные для случая 2.

Базовые алгоритмические структуры

В соответствии со структурной теоремой, изложенной в классической работе итальянцев Бома и Джакопини (1965г.), алгоритм любой программы может быть построен на основе трёх базовых управляющих структур (конструкций):

последовательность, выбор решения и циклическое повторение действий

Последовательность

Последовательная управляющая структура – это непосредственное выполнение одного действия (операции) за другим в порядке их записи сверху вниз.

Выбор решения (разветвление по условию)

Управляющая структура бинарное ветвление – это проверка условия и выбор одного из двух альтернативных действий; выбор осуществляется в зависимости от истинности (true) или ложности (false) проверяемого условия.

Управляющая структура множественное ветвление – это вычисление какого-нибудь константного выражения и, в зависимости от вычисленного значения, выбор одного из множества альтернативных действий.

Циклическое повторение блока действий \ операций

Управляющая структура циклическое повторение – это выполнение повторяющейся последовательности действий (операций), повторение которых осуществляется до тех пор, пока выполняется заданное условие.

Циклическое повторение организуется с программными конструкциями цикл и рекурсия

Существует три разновидности циклов:

Цикл с предусловием – условие повторения проверяется в начале цикла, а затем выполняется тело цикла – действия (операции) входящие в цикл.

Цикл с постусловием – проверка условия осуществляется в конце цикла, после того как будут выполнены действия (операции) входящие в тело цикла.

Цикл с заданным количеством повторений – в заголовке цикла задаётся количество повторений (итераций) цикла: задаётся начальное и конечное значение счётчика итераций цикла и шаг приращения счётчика в каждой итерации, что и является условием повторения.

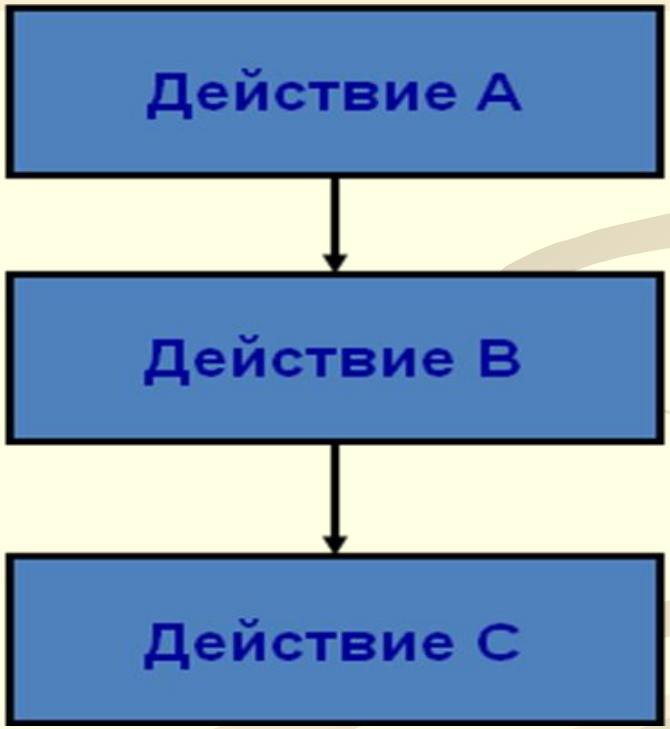
Рекурсия организуется как циклическое повторение специфических подпрограмм рекурсивных функций. Суть рекурсивной функции состоит в том, что она вызывает саму себя, тем самым выполняя циклическое повторение действия\операции. Условие повторения включено в алгоритм, реализуемый рекурсивной функцией.

Представление алгоритмов

Блок-схемы трёх базовых управляющих алгоритмических структур

Последовательность

Непосредственное выполнение одного действия за другим



Представление алгоритмов

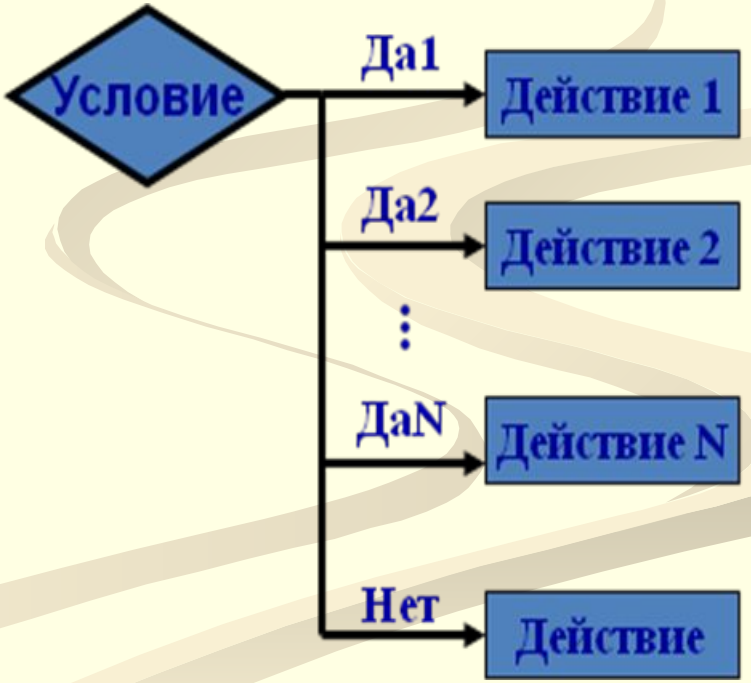
Блок-схемы трёх базовых управляющих алгоритмических структур

Выбор решения

Проверка выполнения условия и выбор одного из альтернативных действий

1. Бинарный выбор – ветвление

2. Множественный выбор



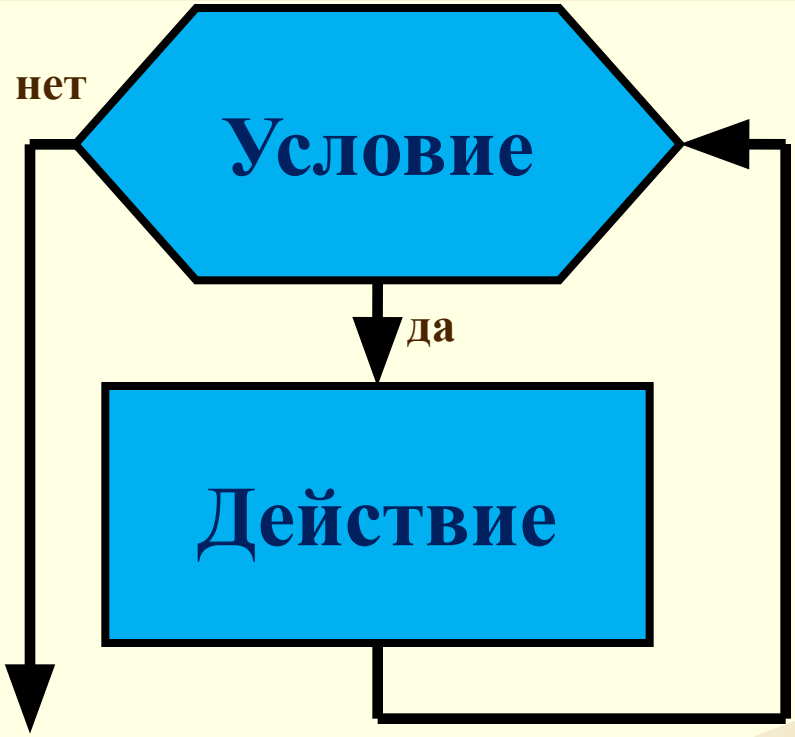
Представление алгоритмов

Блок-схемы трёх базовых управляющих алгоритмических структур

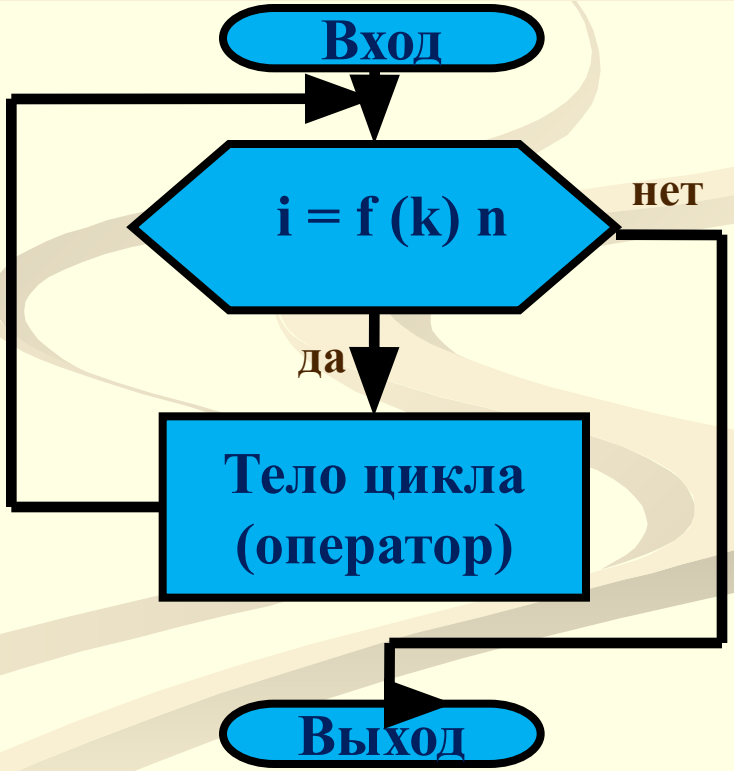
Цикл

Организация повторяющихся действий в соответствии с заданным условием

Общая схема цикла



Цикл с параметром



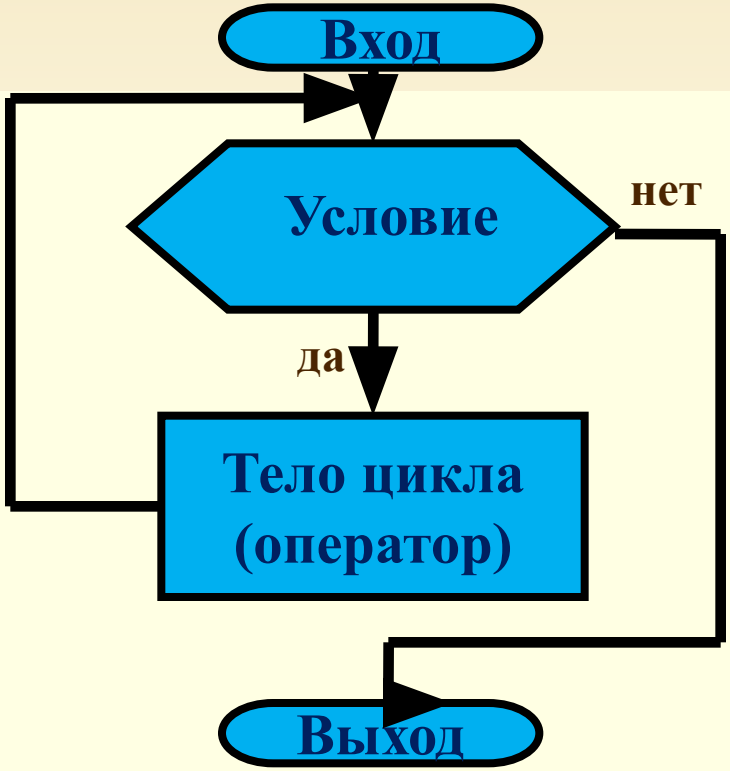
Тело цикла выполняется n-f/k раз

Представление алгоритмов

Блок-схемы трёх базовых управляющих алгоритмических структур

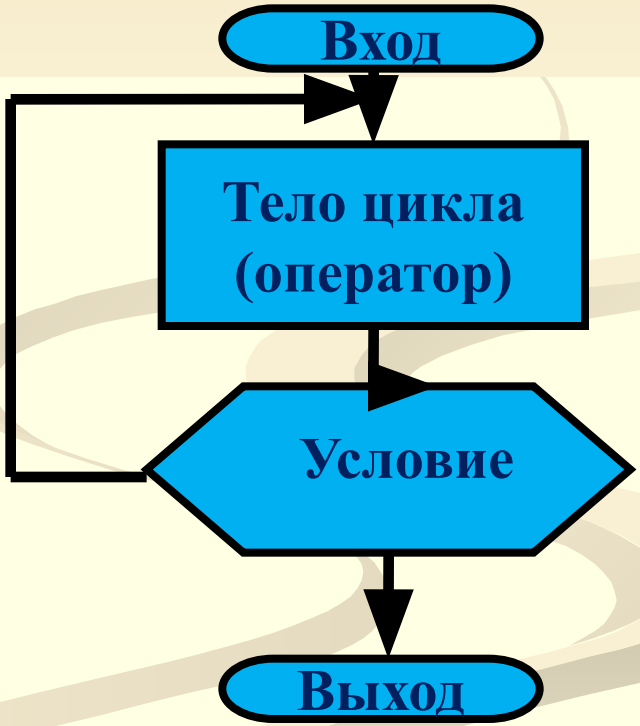
Цикл

Цикл с предусловием



Тело цикла может не выполниться ни разу

Цикл с постусловием



Тело цикла обязательно выполниться хотя бы один раз

Представление алгоритмов

Как раньше было описано, для представления алгоритма могут использоваться различные способы: вербальное описание, графическая схема, таблица решений, описание средствами алгоритмического языка программирования.

Для дальнейшего изучения основ программирования будем использовать представление алгоритмов в виде псевдокода (вербального формализованного описания), в виде блок-схемы (графического описания в совмещённом виде: блок-схема (графическое описание), в графических элементах которой записаны действия/операции в псевдокоде (вербальном формализованном описании). А затем будем преобразовывать блок-схемы в программы на ЯП Pascal и C.

Графические элементы и составление блок-схем были рассмотрены раньше, сейчас рассмотрим подробнее псевдокод.

Псевдокод

Формализация словесного описания в псевдокоде – выбор ключевых слов псевдокода, применяемых для описания алгоритмических операций – производится таким образом, чтобы обеспечить согласование этих ключевых слов с основными операциями компьютера и операциями алгоритмических языков высокого уровня.

Представление алгоритмов

Псевдокод

Основные операции ЯП и ключевые слова псевдокода:

Получение информации

Для описания получения информации компьютером в псевдокоде будут использоваться глаголы:

Прочитать – для чтения входных данных из файла или памяти компьютера;
Получить – для чтения входных данных из буфера клавиатуры компьютера.

Примеры команд (операций) в псевдокоде:

```
Прочитать  имя_ученика, number_1, item_2
Получить  tax_code
```

В этих примерах используется один глагол: **Прочитать** или **Получить**, - за которым следует одно или несколько существительных, указывающих какие именно входные данные должны быть введены в программу.

Выдача информации

Для того чтобы передать информацию из компьютера на внешнее устройство используются глаголы:

Печатать – отправка выходных данных на принтер,
Записать – запись выходных данных в файл,
Вывести – вывод выходных данных на экран,
Запросить – вывод на экран сообщения, требующего ответа пользователя.

Примеры:

```
Печатать "Программа завершена"
Записать имя_заказчика в выходной файл
Вывести  total_tax
Запросить ввод имени ученика
```

(обычно за запросом следует команда Получить).

Например: **Запросить** имя_ученика
Получить имя_ученика

Формат псевдокода позволяет добавлять свои собственные ключевые слова, это обязательно глаголы

Представление алгоритмов

Псевдокод

Основные операции ЯП и ключевые слова псевдокода:

Выполнение математических действий / операций

Для осуществления математических действий в псевдокоде используются знаки математических операций и глаголы:

\+ **Прибавить** – операция сложения,
\- **Вычесть** – операция вычитания,
* **Умножить** – операция умножения,
/ **Делить** – операция деления,

Для выражений используются также:

Вычислить – операция вычисления математического или логического выражения,
( ) **Круглые скобки** – определение порядка вычисления выражения.

Команды псевдокода могут использовать как глаголы, так и знаки операций:
Прибавить number к total или **total = total + number**

Примеры:

Делить total на count
Sales_tax = cost_price * 0.01
Вычислить C = (A – 32) * 5 / 9

Математические выражения в программах на компьютере и алгоритмах выполняются согласно стандартному математическому порядку действий:
первыми вычисляются выражения заключённые в скобки, затем слева направо выполняются операции умножения и деления, а затем слева направо – операции сложения и вычитания.

Результат вычисления математического выражения – число.

Представление алгоритмов

Псевдокод

Основные операции ЯП и ключевые слова псевдокода:

Выполнение логических действий / операций

Для осуществления **логических действий** в псевдокоде используются знаки математических операций и глаголы или местоимения:

?= **Сравнить** – проверить равенство значений двух переменных,
|| **ИЛИ** – логическая операция "или",
&& **И** – логическая операция "и",
> **Больше** – логическая операция "больше",
>= **Больше или равно** – логическая операция "больше или равно",
< **Меньше** – логическая операция "меньше",
<= **Меньше или равно** – логическая операция "меньше или равно",
<> **Не равно** – логическая операция "не равно".

Примеры:

Сравнить total и count
total ?= count
Вычислить (А >= Б || 32 < С)

Результат вычисления логического выражения – Истинно или Ложно.

Представление алгоритмов

Псевдокод

Основные операции ЯП и ключевые слова псевдокода:

Присваивание значений переменным или ячейкам памяти

Псевдокод для присваивания значений употребляется, когда необходимо:

I. Установить исходное значение данных (переменных), используются глаголы

Инициализировать – определить начальное значение переменной при её объявлении в алгоритме или программе,

Установить – изменить текущее значение переменной.

II. Присвоить переменной значение результата обработки данных, используются знак операции или глагол

= **Равно** – операция присваивания переменной, расположенной слева от знака "равно", результата вычисления выражения, расположенного справа от знака "равно".

III. Сохранить значения, которые предполагается использовать позже

Сохранить – записать текущее значение изменяющейся переменной в другую переменную.

Хранить – записать текущее значение переменной в другую переменную (создать копию).

Примеры:

Инициализировать max в ноль
Установить count в ноль
total_price = cost_price + sales_tax
Хранить CustNum в last_CustNum

Представление алгоритмов

Псевдокод

Основные операции ЯП и ключевые слова псевдокода:

Выбор решения – вычисление условия (логического выражения) и выбор из одного из альтернативных действий

Для описания этого действия в псевдокоде используется конструкция с ключевыми словами: **ЕСЛИ**, **ТО**, **ИНАЧЕ**, **ЕСЛИ ВСЁ**.

Синтаксическая конструкция бинарного ветвления/выбора в псевдокоде:

```
ЕСЛИ  условие Р истинно        УСЛОВИЕ – логическое выражение
    ТО
      Выполнение блока действий в случае истинности условия
    ИНАЧЕ
      Выполнение блока действий в случае ложности условия
ЕСЛИ ВСЁ
```

Пример:

```
ЕСЛИ  student_status  равно part_time
    ТО
      Прибавить 1 к part_time_count
    ИНАЧЕ
      Прибавить 1 к full_time_count
ЕСЛИ ВСЁ
```

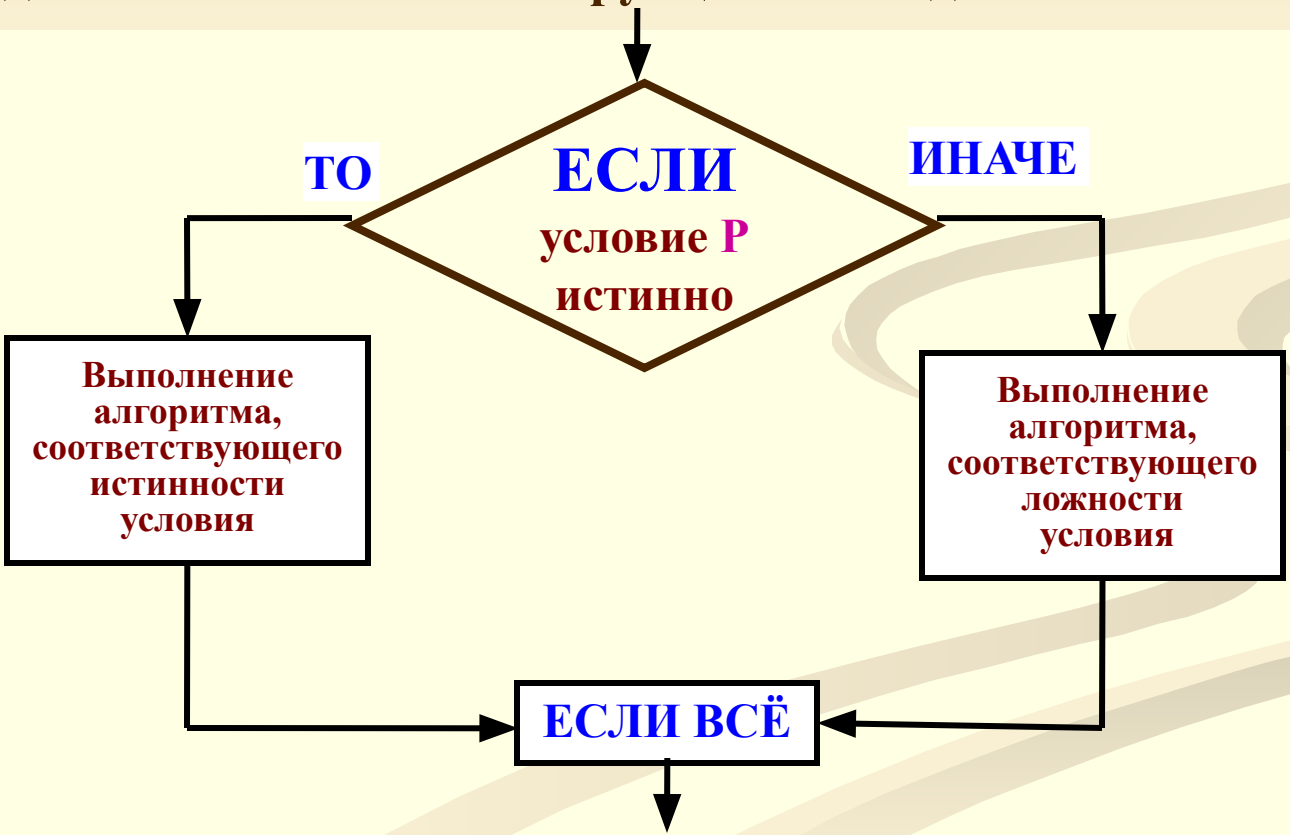
Представление алгоритмов

Псевдокод

Выбор решения

Работа оператора выбор решения состоит в проверке логического условия на истинность и выполнении различных блоков действий в случае истинности или ложности условия.

В виде блок-схемы эта конструкция выглядит так:

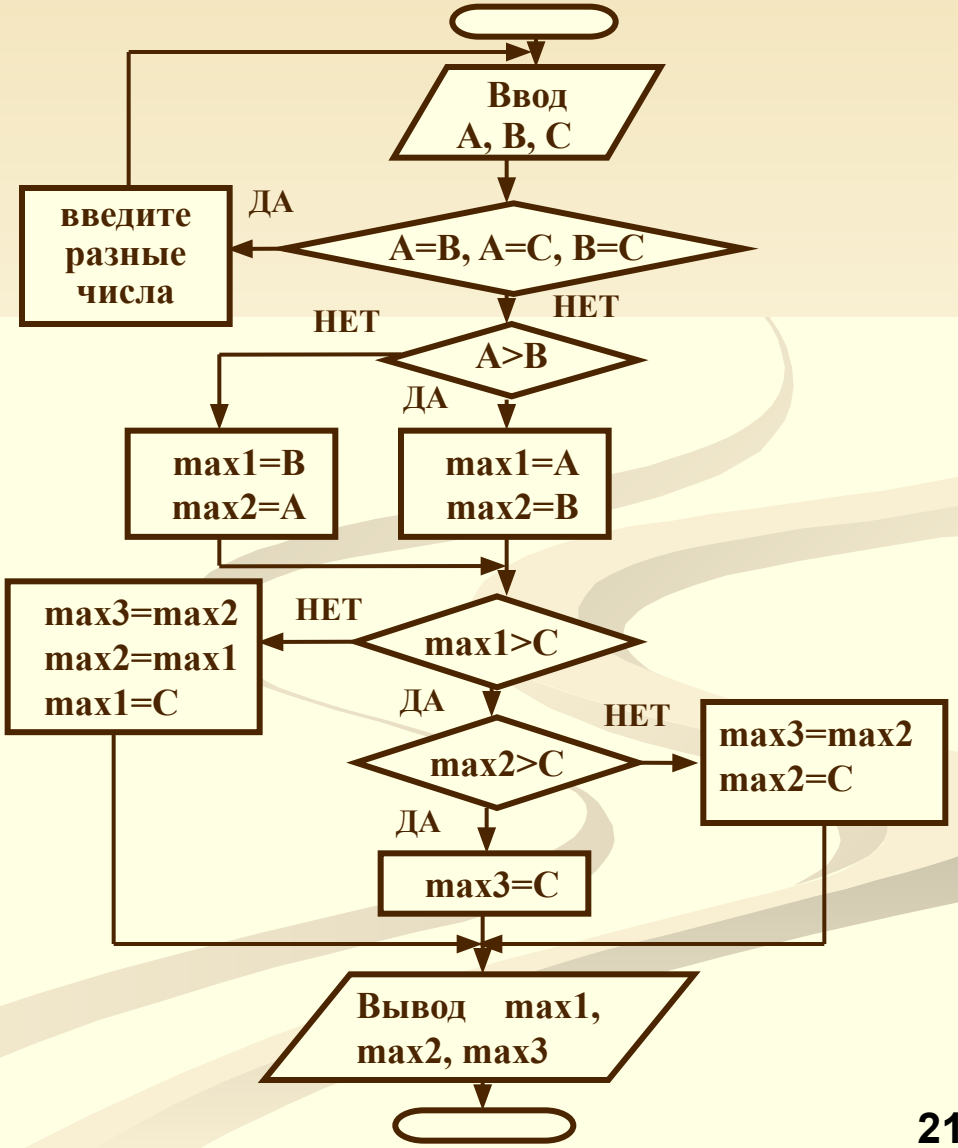


Графическая схема алгоритма

Старый пример

Даны три разные числа A, B, C. Вывести их на печать в порядке убывания.

```
ввести три числа A, B, C
проверить A¬=B, A¬=C, C¬=B
если: A>B
   то:       max1=A, max2=B
   иначе:  max1=B, max2=A
если:  max1>C
   то:
     если:   max2>C
       то:       max3=C
       иначе:  max3=max2
                  max2=C
   иначе:   max3=max2
              max2=max1
              max1=C
вывести на печать max1, max2, max3
```

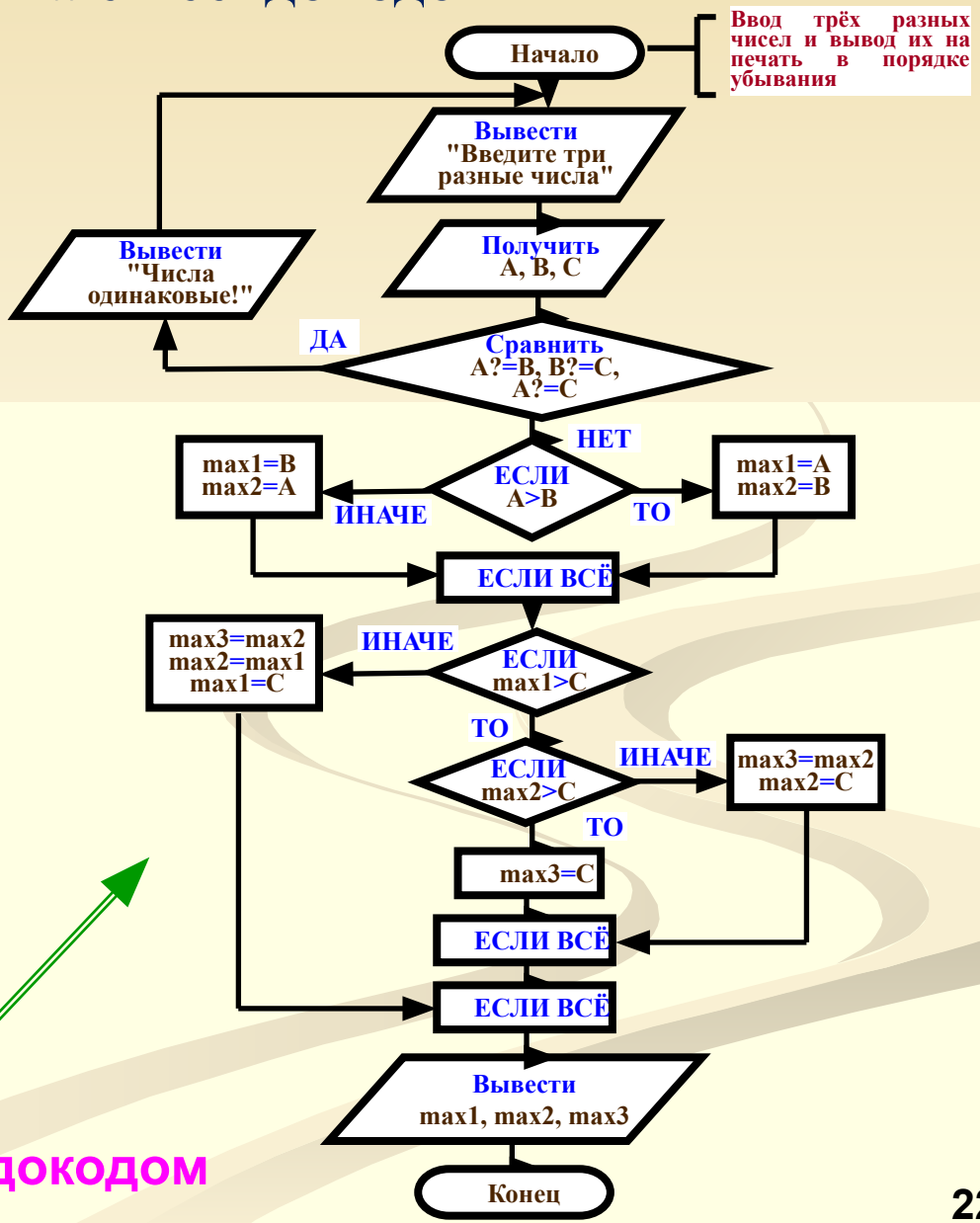


Представление алгоритмов

Блок-схема с псевдокодом

Даны три разные числа A, B, C. Вывести их на печать в порядке убывания.

```
Запросить "Введите три разные числа"
Получить три числа A, B, C
Сравнить A ?= B, A ?= C, C ?= B
 ЕСЛИ A>B
  ТО   max1=A, max2=B
  ИНАЧЕ  max1=B, max2=A
 ЕСЛИ ВСЁ
 ЕСЛИ  max1>C
    ТО
       ЕСЛИ  max2>C
         ТО     max3=C
          ИНАЧЕ  max3=max2
                 max2=C
        ЕСЛИ ВСЁ
     ИНАЧЕ    max3=max2
              max2=max1
              max1=C
 ЕСЛИ ВСЁ
Вывести    max1, max2, max3
```



Нарисовать блок-схему с псевдокодом