Системы программирования и интегрированные среды

Алгоритмические языки и системы программирования

В данный класс программного обеспечения входят средства для создания других программ и программных комплексов.

Программа – это последовательность предписаний (команд), записанных на языке, понятном некоторому исполнителю. В нашем случае исполнителем является процессор компьютера.

Язык программирования – это специально обусловленный набор символов, слов и мнемонических (особым образом организованных и заранее оговоренных сокращений), используемых для записи набора команд (программы), воспринимаемых компьютером.

Языки программирования являются искусственными языками, в них синтаксис и семантика строго определены. Поэтому языки программирования не допускают многозначных и произвольных толкований.

Синтаксис – это набор правил, которые определяют основные внутренние структуры и последовательности символов, допустимых в языке программирования. *Семантика* – это значения языковых единиц (слов, словосочетаний, предложений).

С помощью языка программирования создается не готовая программа, а только ее текст, описывающий ранее разработанный алгоритм. Чтобы получить работающую программу, надо этот текст либо автоматически перевести в машинный код (для этого служат программы-*компиляторы*) и затем использовать отдельно от исходного текста, либо сразу выполнять команды языка, указанные в тексте программы (этим занимаются программы-*интерпретаторы*).

Уровни языков программирования

Разные типы процессоров имеют разные наборы команд. Если язык программирования ориентирован на конкретный тип процессора и учитывает его особенности, то он называется *языком программирования низкого уровня.* В данном случае «низкий уровень» не значит «плохой». Имеется в виду, что операторы языка близки к машинному коду и ориентированы на конкретные команды процессора.

Языком самого низкого уровня является *язык ассемблера,* который просто представляет каждую команду машинного кода, но не в виде чисел, а с помощью символьных условных обозначений, называемых *мнемониками.*

Однозначное преобразование одной машинной инструкции в одну команду ассемблера называется *транслитерацией.* Так как наборы инструкций для каждого модели процессора отличаются, конкретной компьютерной архитектуре соответствует свой язык ассемблера, и написанная на нем программа может быть использована только в этой среде.

С помощью языков низкого уровня создаются очень эффективные и компактные программы, так как разработчик получает доступ ко всем возможностям процессора. С другой стороны, при этом требуется очень хорошо понимать устройство компьютера, затрудняется отладка больших приложений, а результирующая программа не может быть перенесена на компьютер с другим типом процессора. Подобные языки обычно применяют для написания небольших системных приложений, драйверов устройств, модулей стыковки с нестандартным оборудованием, когда важнейшими требованиями становятся компактность, быстродействие и возможность прямого доступа к аппаратным ресурсам. В некоторых областях, например в машинной графике, на языке ассемблера пишутся библиотеки, эффективно реализующие требующие интенсивных вычислений алгоритмы обработки изображений.

Языки программирования высокого уровня значительно ближе и понятнее человеку, нежели компьютеру. Особенности конкретных компьютерных архитектур в них не учитываются, поэтому создаваемые программы на уровне исходных текстов легко переносимы на другие платформы, для которых создан транслятор этого языка. Разрабатывать программы на языках высокого уровня с помощью понятных и мощных команд значительно проще, а ошибок при создании программ допускается гораздо меньше.

Поколения языков программирования

Языки программирования принято делить на пять *поколений.* В первое поколение входят языки, созданные в начале 50-х годов, когда первые компьютеры только появились на свет. Это был первый язык ассемблера, созданный по принципу «одна инструкция — одна строка».

Расцвет второго поколения языков программирования пришелся на конец 50-х — начало 60-х годов. Тогда был разработан символический ассемблер, в котором появилось понятие переменной. Он стал первым полноценным языком программирования. Благодаря его возникновению заметно возросли скорость разработки и надежность программ.

Появление третьего поколения языков программирования принято относить к 60-м годам. В это время родились универсальные языки высокого уровня, с их помощью удается решать задачи из любых областей. Такие качества новых языков, как относительная простота, независимость от конкретного компьютера и возможность использования мощных синтаксических конструкций, позволили резко повысить производительность труда программистов. Понятная большинству пользователей структура этих языков привлекла к написанию небольших программ значительное число специалистов из некомпьютерных областей. Подавляющее большинство языков этого поколения успешно применяется и сегодня.

С начала 70-х годов по настоящее время продолжается период языков четвертого поколения. Эти языки предназначены для реализации крупных проектов, повышения их надежности и скорости создания. Они обычно ориентированы на специализированные области применения, где хороших результатов можно добиться, используя не универсальные, а проблемно-ориентированные языки, оперирующие конкретными понятиями узкой предметной области.

Рождение языков пятого поколения произошло в середине 90-х годов. К ним относятся также системы автоматического создания прикладных программ с помощью визуальных средств разработки, без знания программирования. Главная идея, которая закладывается в эти языки, — возможность автоматического формирования результирующего текста на универсальных языках программирования (который потом требуется откомпилировать). Инструкции же вводятся в компьютер в максимально наглядном виде с помощью методов, наиболее удобных для человека, не знакомого с программированием.

Обзор языков программирования высокого уровня

Fortran (Фортран). Это первый компилируемый язык, созданный Джимом Бэкусом в 50-е годы. Программисты, разрабатывавшие программы исключительно на ассемблере, выражали серьезное сомнение в возможности появления высокопроизводительного языка высокого уровня, поэтому основным критерием при разработке компиляторов Фортрана являлась эффективность исполняемого кода. Хотя в Фортране впервые был реализован ряд важнейших понятий программирования, удобство создания программ было принесено в жертву возможности получения эффективного машинного кода.

Однако для этого языка было создано огромное количество библиотек, начиная от статистических комплексов и кончая пакетами управления спутниками, поэтому Фортран продолжает активно использоваться во многих организациях. Имеется стандартная версия Фортрана *HPF* (High Performance Fortran) для параллельных суперкомпьютеров со множеством процессоров.

Cobol (Кобол). Это компилируемый язык для применения в экономической области и решения бизнес-задач, разработанный в начале 60-х годов. Он отличается большой «многословностью» — его операторы иногда выглядят как обычные английские фразы. В Коболе были реализованы очень мощные средства работы с большими объемами данных, хранящимися на различных внешних носителях. На этом языке создано очень много приложений, которые активно эксплуатируются и сегодня. Достаточно сказать, что наибольшую зарплату в США получают программисты на Коболе.

Algol (Алгол). Компилируемый язык, созданный в 1960 году. Он был призван заменить Фортран, но из-за более сложной структуры не получил широкого распространения. В 1968 году была создана версия Алгол 68, по своим возможностям и сегодня опережающая многие языки программирования, однако из-за отсутствия достаточно эффективных компьютеров для нее не удалось своевременно создать хорошие компиляторы.

Pascal (Паскаль). Язык Паскаль, созданный в конце 70-х годов основоположником множества идей современного программирования Никлаусом Виртом, во многом напоминает Алгол, но в нем ужесточен ряд требований к структуре программы и имеются возможности, позволяющие успешно применять его при создании крупных проектов.

Basic (Бейсик). Для этого языка имеются и компиляторы, и интерпретаторы, а по популярности он занимает первое место в мире. Он создавался в 60-х годах в качестве учебного языка и очень прост в изучении.

C (Си). Данный язык был создан в лаборатории Bell и первоначально не рассматривался как массовый. Он планировался для замены ассемблера, чтобы иметь возможность создавать столь же эффективные и компактные программы, и в то же время не зависеть от конкретного типа процессора.

Си во многом похож на Паскаль и имеет дополнительные средства для прямой работы с памятью *(указатели).* На этом языке в 70-е годы написано множество прикладных и системных программ и ряд известных операционных систем (Unix).

C++ (Си++). Си++ — это объектно-ориентированное расширение языка Си, созданное Бьярном Страуструпом в 1980 году. Множество новых мощных возможностей, позволивших резко повысить производительность программистов, наложилось на унаследованную от языка Си определенную низкоуровневость, в результате чего создание сложных и надежных программ потребовало от разработчиков высокого уровня профессиональной подготовки.

Java (Джава, Ява). Этот язык был создан компанией Sun в начале 90-х годов на основе Си++. Он призван упростить разработку приложений на основе Си++ путем исключения из него всех низкоуровневых возможностей. Но главная особенность этого языка — компиляция не в машинный код, а в платформно-независимый байт-код (каждая команда занимает один байт). Этот байт-код может выполняться с помощью интерпретатора — виртуальной Java-машины JVM (Java Virtual Machine), версии которой созданы сегодня для любых платформ. Благодаря наличию множества Java-машин программы на Java можно переносить не только на уровне исходных текстов, но и на уровне двоичного байт-кода, поэтому по популярности язык Ява сегодня занимает второе место в мире после Бейсика.

Особое внимание в развитии этого языка уделяется двум направлениям: поддержке всевозможных мобильных устройств и микрокомпьютеров, встраиваемых в бытовую технику (технология Jini) и созданию платформно-независимых программных модулей, способных работать на серверах в глобальных и локальных сетях с различными операционными системами (технология Java Beans). Пока основной недостаток этого языка — невысокое быстродействие, так как язык Ява интерпретируемый.

Языки программирования для Интернета

С активным развитием глобальной сети было создано немало реализации популярных языков программирования, адаптированных специально для Интернета. Все они отличаются характерными особенностями: языки являются интерпретируемыми, интерпретаторы для них распространяются бесплатно, а сами программы — в исходных текстах. Такие языки называют скрипт-языками.

HTML. Общеизвестный язык для оформления документов. Он очень прост и содержит элементарные команды форматирования текста, добавления рисунков, задания шрифтов и цветов, организации ссылок и таблиц. Все Web-страницы написаны на языке HTML или используют его расширения.

Perl. В 80-х годах Ларри Уолл разработал язык Perl. Он задумывался как средство эффективной обработки больших текстовых файлов, генерации текстовых отчетов и управления задачами. По мощности Perl значительно превосходят языки типа Си. В него введено много часто используемых функций работы со строками, массивами, всевозможные средства преобразования данных, управления процессами, работы с системной информацией и др.

Tcl/Tk. В конце 80-х годов Джон Аустираут придумал популярный скрипт-язык Tcl и библиотеку Tk. В Tcl он попытался воплотить видение идеального скрипт-языка. Tcl ориентирован на автоматизацию рутинных процессов и состоит из мощных команд, предназначенных для работы с абстрактными нетипизированными объектами. Он независим от типа системы и при этом позволяет создавать программы с графическим интерфейсом.

VRML. В 1994 году был создан язык VRML для организации виртуальных трехмерных интерфейсов в Интернете. Он позволяет описывать в текстовом виде различные трехмерные сцены, освещение и тени, текстуры (покрытия объектов), создавать свои миры, путешествовать по ним, «облетать» со всех сторон, вращать в любых направлениях, масштабировать, регулировать освещенность и т. д.

Интегрированные системы программирования

В самом общем случае для создания программы ва выбранном языке программирования нужно иметь следующие компоненты:

1. *Текстовый редактор.* Так как текст программы записывается с помощью ключевых слов, обычно происходящих от слов английского языка, и набора стандартных символов для записи всевозможных операций, то формировать этот текст можно в любом редакторе, получая в итоге текстовый файл с исходным текстам программы. Подобные редакторы созданы для всех популярных языков и дополнительно могут автоматически проверять правильность синтаксиса программы непосредственно во время ее ввода.

2. Исходный текст с помощью *программы-компилятора* переводится в машинный код. Если обнаружены синтаксические ошибки, то результирующий код создан не будет.

На этом этапе уже возможно получение готовой программы, но чаще всего в ней не хватает некоторых компонентов, поэтому компилятор обычно выдает промежуточный *объектный код* (двоичный файл, стандартное расширение .OBJ).

3. *Редактор связей*. Исходный текст большой программы состоит, как правило, из нескольких модулей (файлов с исходными текстами), потому что хранить все тексты в одном файле неудобно — в них сложно ориентироваться. Каждый модуль компилируется в отдельный файл с объектным кодом, которые затем надо объединить в одно целое.

Кроме того, к ним надо добавить машинный код подпрограмм, реализующих различные стандартные функции (например вычисляющих математические функции sin или ln). Такие функции содержатся в библиотеках (файлах со стандартным расширением .LIB), которые поставляются вместе с компилятором. Сгенерированный код модулей и подключенные к нему стандартные функции надо не просто объединить в одно целое, а выполнить такое объединение с учетом требований операционной системы, то есть получить на выходе программу, отвечающую определенному формату.

Объектный код обрабатывается специальной программой — *редактором связей* или сборщиком, который выполняет связывание объектных модулей и машинного кода стандартных функций, находя их в библиотеках, и формирует на выходе работоспособное приложение — *исполнимый код* для конкретной платформы.

Если по каким-то причинам один из объектных модулей или нужная библиотека не обнаружены (например, неправильно указан каталог с библиотекой), то сборщик сообщает об ошибке и готовой программы не получается.

4. *Исполнимый код* — это законченная программа, которую можно запустить на любом компьютере, где установлена операционная система, для которой эта программа создавалась. Как правило, итоговый файл имеет расширение .EXE или .COM.

Как правило, в стандартную поставку входят как минимум три последних компонента, но хорошая *интегрированная система* включает в себя и специализированный текстовый редактор, причем почти все этапы создания программы в ней автоматизированы: после того как исходный текст введен, его компиляция и сборка выполняются одним нажатием клавиши. Это очень удобно, так как не требует ручной настройки множества параметров запуска компилятора и редактора связей, указывания им нужных файлов вручную и т.д. Процесс компиляции обычно демонстрируется на экране: показывается, сколько строк исходного текста откомпилировано, или выдаются сообщения о найденных ошибках.

В современных интегрированных системах имеется еще один компонент — *отладчик*, который позволяет анализировать работу программы во время ее выполнения. С его помощью можно последовательно выполнять отдельные операторы исходного текста *по шагам,* наблюдая при этом, как меняются значения различных переменных. Без отладчика разработать крупное приложение очень сложно.

Основные системы программирования

Из универсальных языков программирования сегодня наиболее популярны следующие:

Бейсик (Basic) — для освоения требует начальной подготовки (общеобразовательная школа);

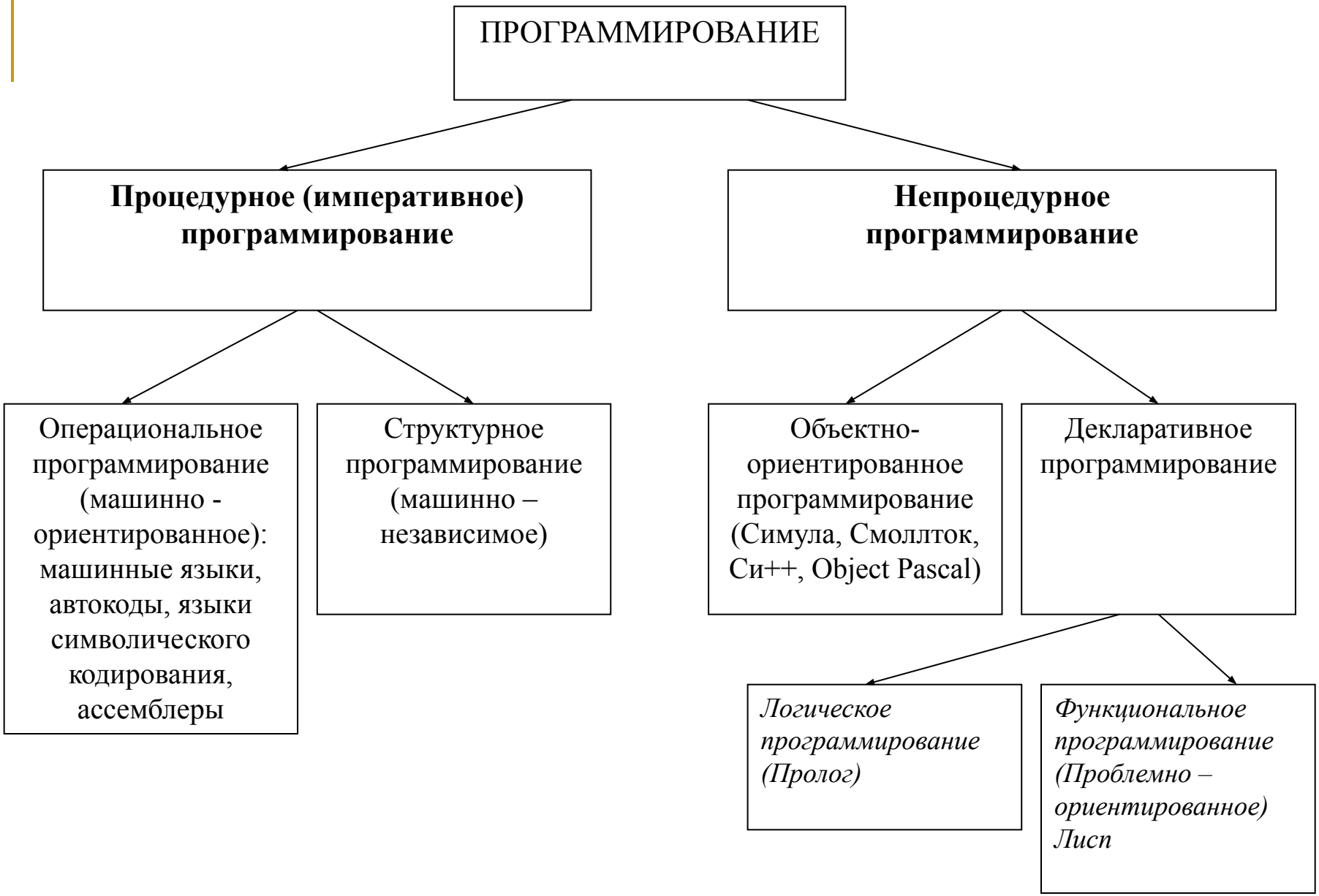
Паскаль (Pascal) — требует специальной подготовки (школы с углубленным изучением предмета и общетехнические ВУЗы);

Си++ (C++), Ява (Java) — требуют профессиональной подготовки (специализированные средние и высшие учебные заведения).

Для каждого из этих языков программирования сегодня имеется немало систем программирования, выпускаемых различными фирмами и ориентированных на различные модели ПК и операционные системы. Наиболее популярны следующие визуальные среды быстрого проектирования программ для Windows:

- Basic: Microsoft Visual Basic
- Pascal: Borland Delphi
- C++: Borland C++ Bulider
- Java: Symantec Café.

Для разработки серверных и распределенных приложений можно использовать систему программирования Microsoft Visual C++, продукты фирмы Inprise под маркой Borland, практически любые средства программирования на Java.


ПРОГРАММИРОВАНИЕ
Процедурное (императивное) программирование
Непроцедурное программирование
Операциональное программирование (машинно - ориентированное): машинные языки, автокоды, языки символического кодирования, ассемблеры
Структурное программирование (машинно – независимое)
Объектно-ориентированное программирование (Симула, Смоллток, Си++, Object Pascal)
Декларативное программирование
Логическое программирование (Пролог)
Функциональное программирование (Проблемно – ориентированное) Лисп

МОДЕЛИРОВАНИЕ—ЭТО:

- построение моделей реально существующих объектов (предметов, явлений, процессов);
- замена реального объекта его подходящей копией;
- исследование объектов познания на их моделях;

Моделирование представляет собой один из основных методов познания и является неотъемлемым элементом любой целенаправленной деятельности.

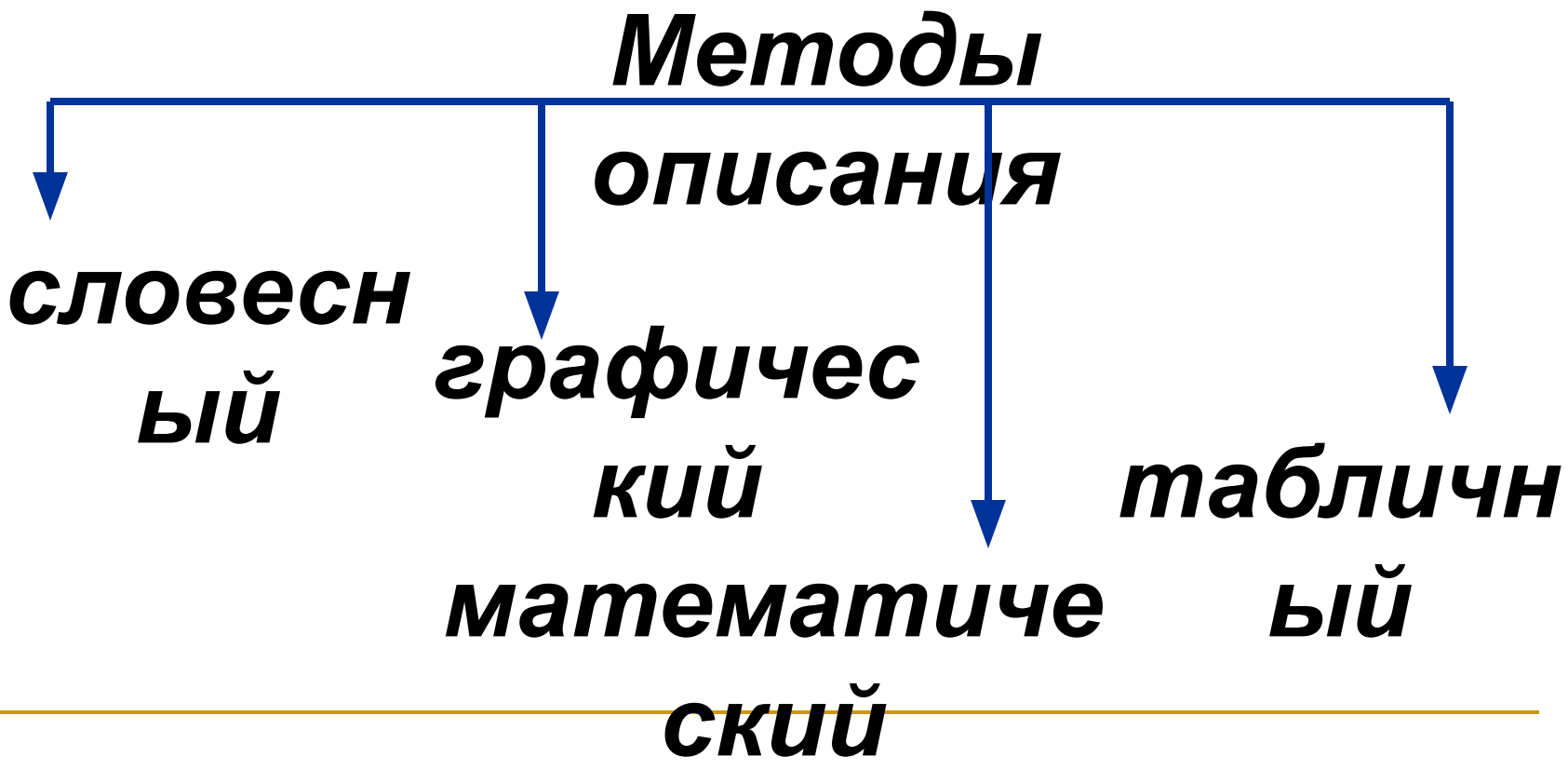
Моделирование - это универсальный метод получения, описания и использования знаний,,

Что такое модель?

Модель (фр.modele, ит.modello, лат.modulus-мера, образец)-это:

- Некоторое упрощенное подобие реального объекта;
- Воспроизведение предмета в уменьшенном или увеличенном виде (макет);
- Схема, изображение или описание какого-либо явления или процесса в природе и обществе;
- Физический или информационный аналог объекта, функционирование которого по определенным параметрам подобно функционированию реального объекта;
- Некий объект-заместитель, который в определенных условиях может заменять объект-оригинал, воспроизводя интересующие нас его свойства и характеристики, причем имеет существенные преимущества или удобства(наглядность, обозримость, доступность испытаний, легкость оперирования с ним и пр.);
- Новый объект, который отражает некоторые стороны изучаемого объекта или явления, существенные с точки зрения моделирования;
- Новый объект(реальный, информационный или воображаемый), отличный от исходного, который обладает существенными для целей моделирования свойствами и в рамках этих целей полностью заменяет исходный объект.

Какие существуют методы описания?



Классы моделей

Все многообразие моделей делится на три класса:

- Материальные (натуральные) модели (некие реальные предметы-макеты, муляжи, эталоны) уменьшенные или увеличенные копии, воспроизводящие внешний вид моделируемого объекта, его структуру(глобус, модель кристаллической решетки) или поведение (велотренажер);
- Воображаемые модели (геометрическая точка, бесконечность, математический маятник);
- Информационные модели – описания моделируемого объекта на одном из языков кодирования информации (словесное описание, схемы, чертежи, карты, рисунки, научные формулы, программы и т. д.).

Информационная модель — система сигналов, свидетельствующих о динамике объекта управления, условиях внешней среды и состоянии самой системы управления. В качестве информационной модели могут служить наглядные изображения (фото, кино, видео), знаки (текст, знаковое табло), графические модели (график, чертеж, блок–схема) и комбинированные изображения (мнемосхема, карта).

Модели бывают:

ПО НАЗНАЧЕНИЮ:

- Познавательная(форма организации и представления знаний, средство соединения новых и старых знаний. Познавательная модель, как правило, подгоняется под реальность и является теоретической моделью.)
- Прагматическая(средство организации практических действий, рабочего представления целей системы для ее управления. Это, как правило, прикладная модель.)
- Инструментальная(средство построения, исследования, использования первых двух моделей.)

ПО УРОВНЮ МОДЕЛИРОВАНИЯ:

♪Эмпирическими – на основе эмпирических фактов, зависимостей;

♪Теоретическими – на основе математических описаний;

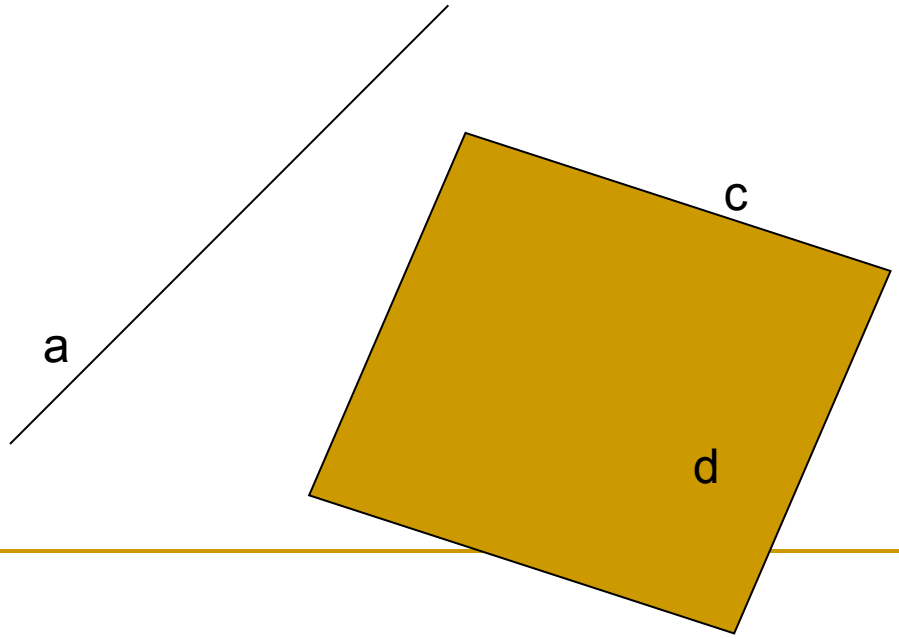
♪Смешанными, полуэмпирическими .

Типы моделей.

- Статическая:если среди параметров, участвующих в описании модели, нет временного параметра.
- Динамическая:если среди параметров модели есть временной параметр.
- Дискретная:если она описывает поведение системы только в дискретные моменты времени.
- Теоретико-множественная:если представима с помощью некоторых множеств и отношений принадлежности им и между ними.
- Логическая:если она представима предикатами, логическими функциями.
- Геометрическая.
- Игровая.
- Алгоритмическая.

Примеры:

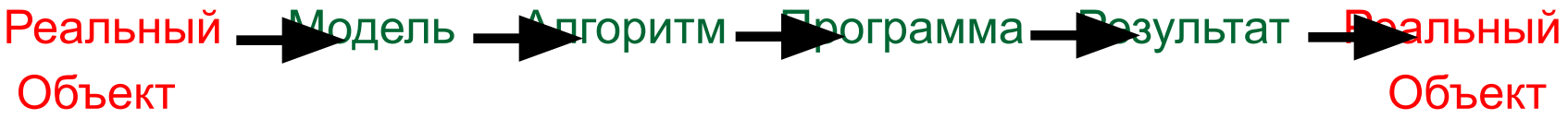
1. F=m*a – *статическая модель.*
2. *Z=X Λ У – логическая модель.*
3. *Макет дома, параллелограмм, прямая линия – геометрические модели.*



Основные свойства моделей.

- Конечность – модель отображает оригинал лишь в конечном числе его отношений.
- Упрощенность – модель отображает только существенные стороны объекта.
- Приблизительность.
- Адекватность.
- Наглядность, обозримость.
- Доступность и технологичность.
- Информативность, т.е. модель должна содержать достаточную информацию о системе.
- Сохранение информации, содержавшейся в оригинале.
- Полнота.
- Устойчивость.
- Замкнутость – модель учитывает и отображает замкнутую систему необходимых основных гипотез, связей и отношений.

С точки зрения информатики
решение любой задачи описывается так:



Основные принципы формализации

- разработка неформального описания модели (словесное описание существенных для рассматриваемой задачи характеристик изучаемого объекта и связей между ними);
- составление формализованного описания на некотором языке кодирования (с использованием математических соотношений и текстов);
- реализация формализованного описания в виде программы на некотором языке программирования.

АЛГОРИТМИЗАЦИЯ

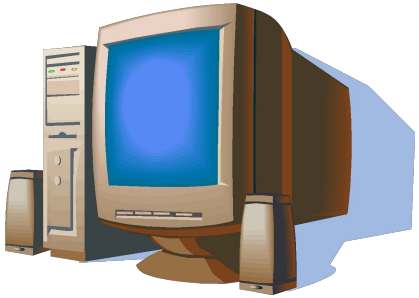
исполнение которых приводит к решению поставленной задачи за конечное число шагов

- Появление алгоритмов связывают с зарождением математики. Более 1000 лет назад (в 825 году) учёный из города Хорезма Абдулла (или Абу Джафар) Мухаммед бен Муса аль – Хорезми создал книгу по математике, в которой описал способы выполнения арифметических действий над многозначными числами. Само слово «алгоритм» возникло в Европе после перевода на латынь книги этого среднеазиатского математика, в которой его имя писалось как «Алгоритми»

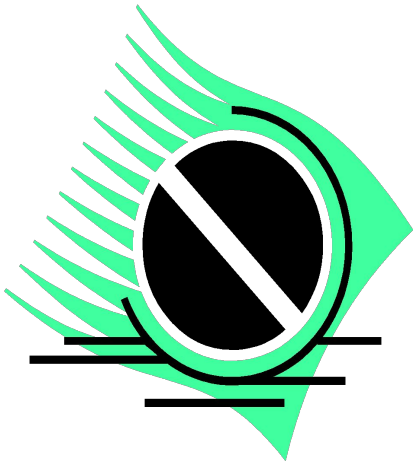
Исполнитель алгоритма

Исполнитель - это объект, умеющий выполнять определенный набор действий. (человек, животное, робот, компьютер).

- Человек – может выполнять алгоритм формально, не вникая в содержание поставленной задачи, а только строго выполняя последовательность действий, предусмотренную алгоритмом
- Компьютер – автоматическое исполнение алгоритма, представленного в виде программы (алгоритме, записанного на «понятном» языке программирования)



Исполнителя характеризуют:



- Среда – «место обитания» исполнителя
- Система команд – каждый Исполнитель может выполнять команды только из некоторого строго заданного списка системы команд исполнителя (СКИ). Для каждой команды должны быть заданы условия применимости (в каких состояниях среды может быть выполнена команда) и описаны результаты выполнения команды.
- Элементарное действие , которое совершает исполнитель после вызова команды
- Отказ – возникает, если команда вызывается при недопустимом для неё состоянии среды

Свойства алгоритма

- **Понятность** – исполнитель алгоритма должен знать, как его выполнять
- **Определённость** – каждое правило алгоритма должно быть чётким, однозначным и не оставлять места для произвола
- **Результативность** – состоит в том, что алгоритм должен приводить к решению задачи за конечное число шагов
- **Массовость** – означает, что алгоритм решения задачи разрабатывается в общем виде, т.е. он должен быть применим для некоторого класса задач, различающихся лишь исходными данными
- **Дискретность** – алгоритм должен представлять процесс решения задачи как последовательное выполнение простых шагов

Способы записей алгоритмов

- **Словесный способ** – представляет собой описание последовательных этапов обработки данных. Алгоритм задаётся в произвольном изложении на естественном языке. Словесный способ не имеет широкого распространения, т.к. такие описания страдают многословностью записей и допускают неоднозначности толкования отдельных предписаний.
- **Графический способ** – изображения из графических символов (блок-схемы)
- **Псевдокоды** (полуформализованные описания алгоритмов на условном алгоритмическом языке, включающие как элементы языка программирования, таки фразы естественного языка, общепринятые математические обозначения и др.)

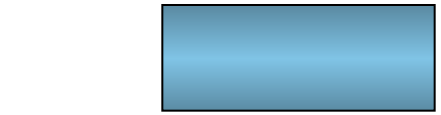
Язык блок *– схем является одним из способов символической записи алгоритмов.*

- **Структурная блок-схема** – схема алгоритма – графическое изображение алгоритма в виде схемы, связанных между собой с помощью стрелок блоков.
- **Стрелки** – линии перехода.
- **Блок** – графический символ, каждый из которых соответствует одному шагу алгоритма. Внутри блока дается описание соответствующего действия.

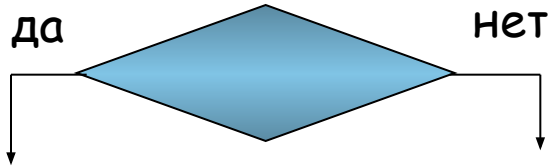
Графическое изображение алгоритма широко используется перед программированием задачи вследствие его наглядности, так как зрительное восприятие облегчает процесс написания программы, его корректировки при возможных ошибках, осмысление процесса обработки информации.

Основные алгоритмические структуры

Блочные символы



Вычислительное действие или последовательность действий

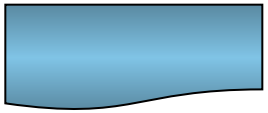


Проверка условий

Начало, конец алгоритма, вход и выход в подпрограмму

Вычисления по подпрограмме

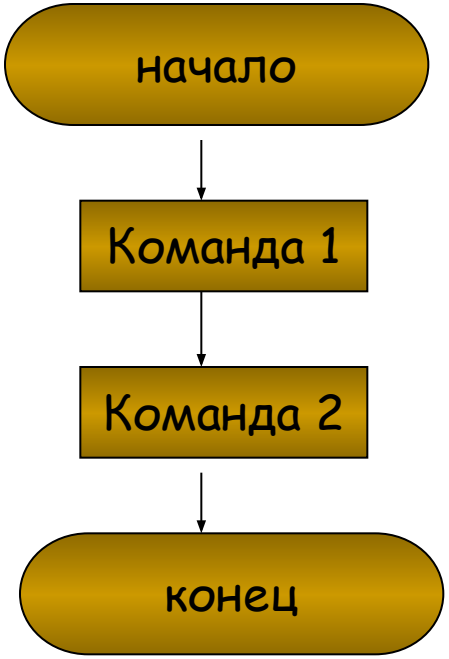
Ввод-вывод в общем виде



Вывод результатов на печать

Линейная структура алгоритма

Линейным называется алгоритм, в котором команды выполняются последовательно друг за другом

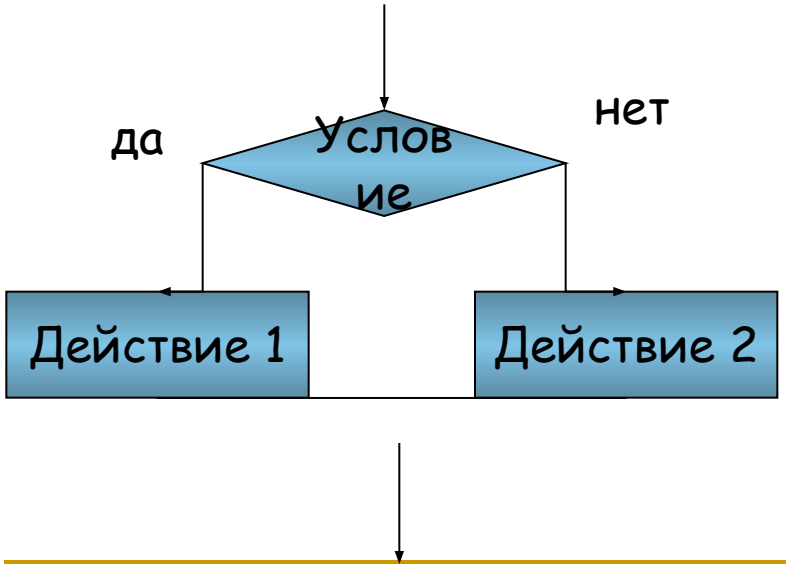


Алгоритмическая структура «ветвление»

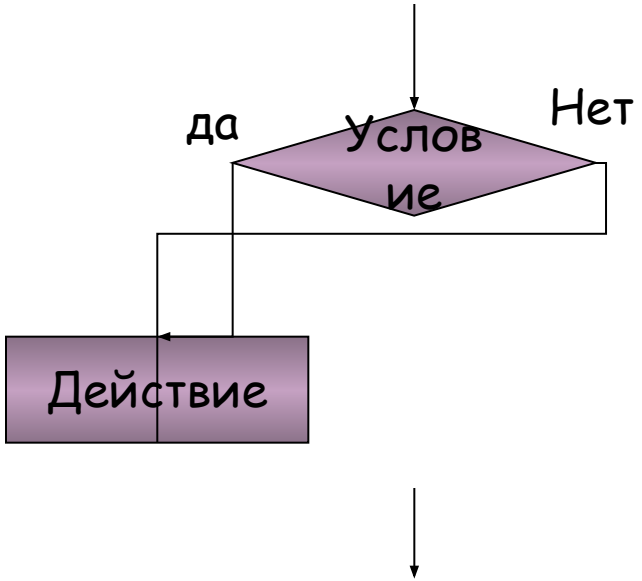
Разветвляющийся алгоритм – алгоритм, в котором проверяется условие, в зависимости от которого выполняется то или иное действие.

Условие – выражение, находящееся между словами «если» и словом «то» и принимающее значение «истина» или «ложь»

Полное ветвление

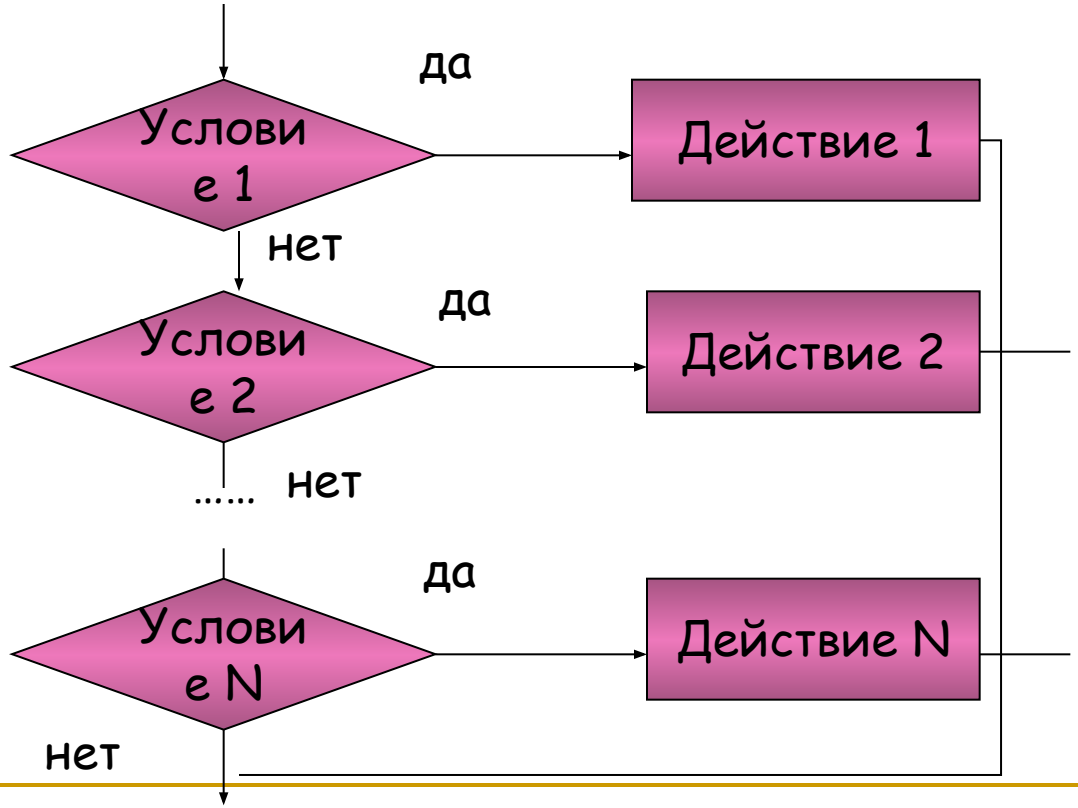


Неполное ветвление



Алгоритмическая структура «выбор»

В алгоритмической структуре «выбор» выполняется одна из нескольких последовательностей команд при истинности соответствующего условия

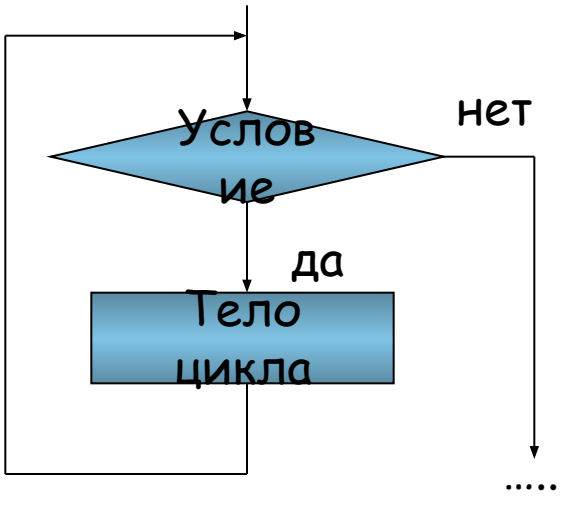


Алгоритмическая структура «цикл»

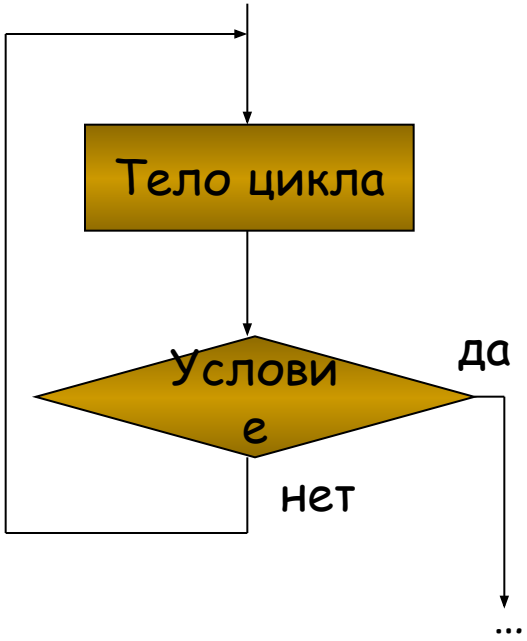
Циклический алгоритм – описание действий, которые должны повторяться указанное число раз или пока не выполнено заданное условие

Перечень повторяющихся действий называется **телом цикла**

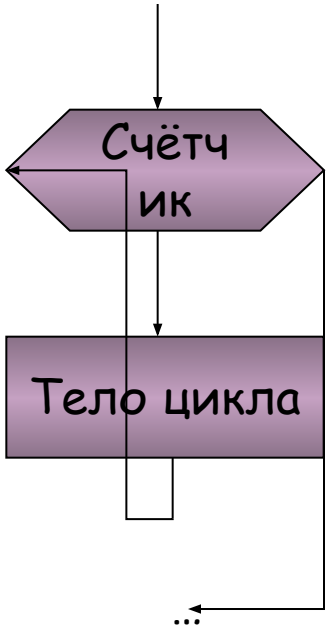
Цикл с предусловием



Цикл с постусловием



Цикл с параметром



Этапы решения задачи на компьютере

Первый этап – **постановка задачи**.

Второй этап – **математическое моделирование**.

Третий этап – **алгоритмизация задачи**.

Четвертый этап – **программирование**.

Пятый этап – **ввод программы и исходных данных в компьютер**.

Шестой этап – **тестирование и отладка программы.**

Седьмой этап – **выполнение отлаженной программы и анализ результатов**.

Пример №1

Составьте алгоритм вычисления выражения y=2x+в, x=5, в=5.

На естественном языке:

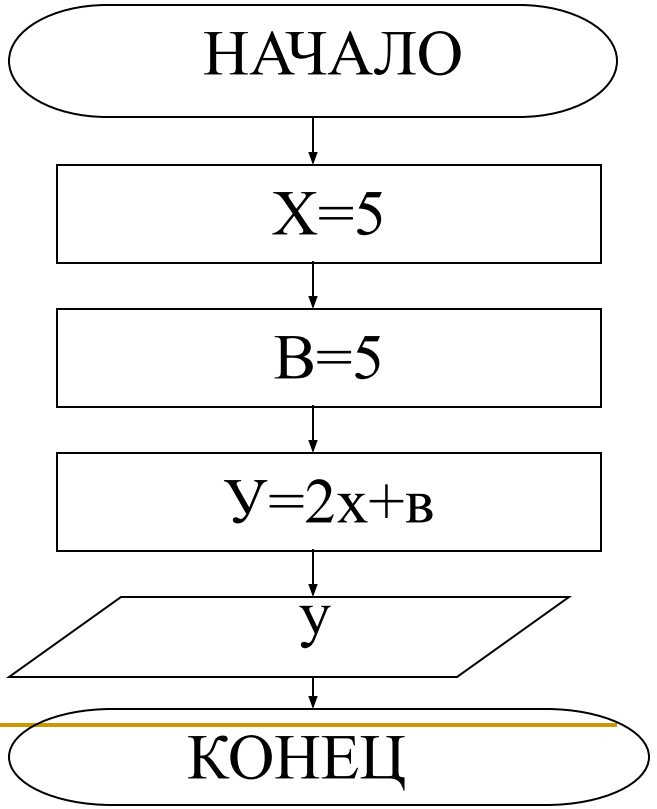
1.x=5

2. B=5

3.y=2x+в

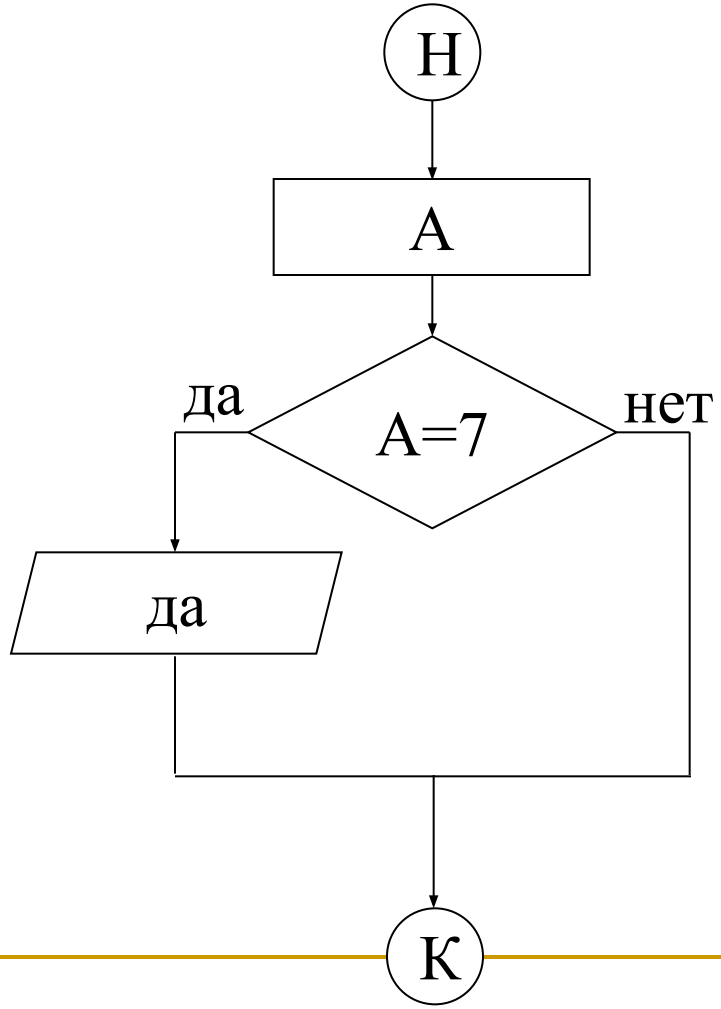
4. Напечатать y

На языке блок-схем:



Пример №2.

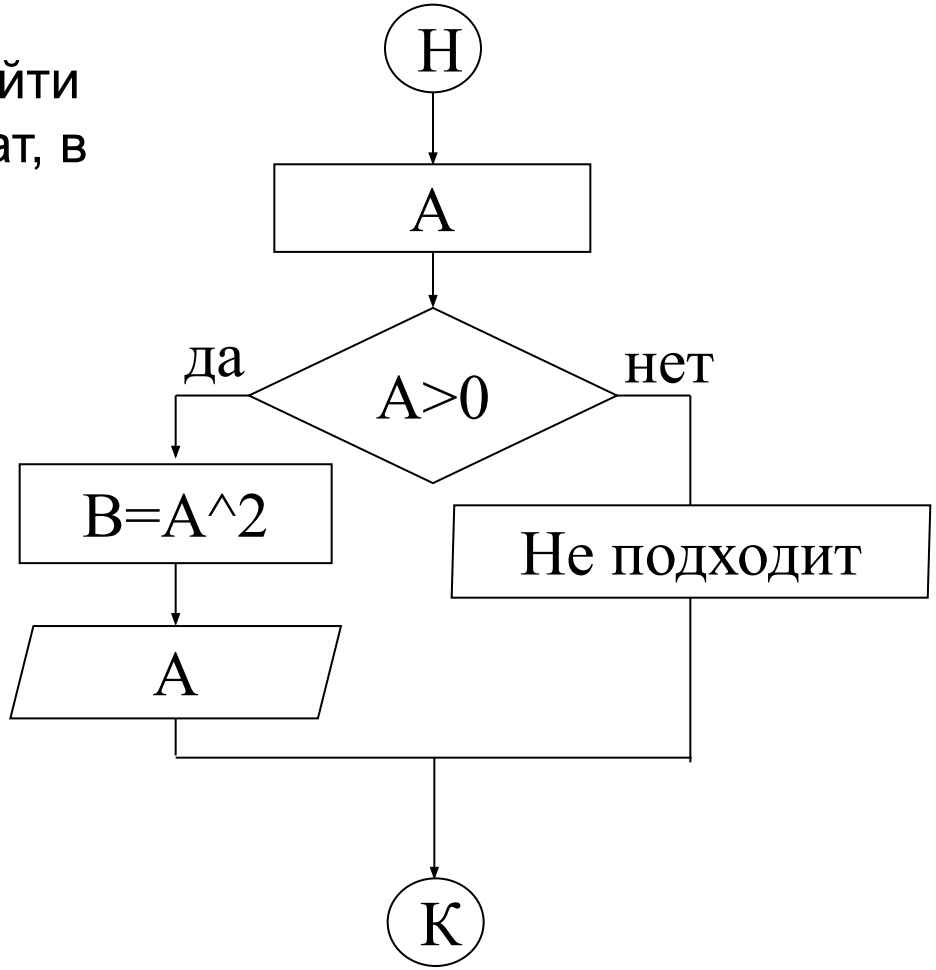
Проверить, равно ли введенное число цифре 7, и в случае равенства выдать сообщение «Да».



1. Ввести число
2. Проверка условия А=7
3. Если да, то печать «Да»
4. Конец
5. Иначе конец

Пример №3

Если число положительное, то найти его квадрат и вывести результат, в противном случае вывести сообщение «Не подходит». Составить программу двумя способами: с условным и безусловным переходом



Пример №4

Составить программу и блок-схему вывода чисел от 5 до 10.

1. Для каждого числа от 5 до 10
2. Повторять действие
3. Печать этого числа
4. Конец