Дисциплина «Информатика»
Специальность №08080165 «Прикладная информатика (в экономике)»
Институт информатики, инноваций и бизнес систем
Кафедра информатики, инженерной и компьютерной графики
Старший преподаватель Молоков К.А.

Элементы алгоритмизации и программирования

Алгоритмизация: алгоритмы и способы их описания, составление алгоритмов на языке блок-схем, базовые управляющие конструкции алгоритмов. Понятие языка высокого уровня. Синтаксис и семантика. Полный цикл работы с программой. Выполнение вычислительных операций. Циклические конструкции. Работа с символьными и строковыми переменными. Записи и множества. Обработка массивов данных. Процедуры и функции. Построение графических изображений. Операции с файлами. Визуальное программирование.

ВГУЭС

Этапы решения задач

Для успешного использования ЭВМ в своей профессиональной деятельности пользователь должен уметь формулировать задачи, разрабатывать алгоритмы их решения, записывать алгоритмы на языке, понятном ЭВМ.

Этапы подготовки и решения реальных задач включают:

- постановку задачи
- физическое моделирование
- математическое или информационное моделирование
- алгоритмизацию задачи
- разработку программы
- тестирование и отладка программ
- анализ результатов

Алгоритмы и способы их описания

Алгоритм — точный порядок действий, определяющий процесс, ведущий от исходных данных к искомому результату.

Алгоритм — это конечная последовательность однозначных предписаний, исполнение которых позволяет с помощью конечного числа шагов получить решение задачи, однозначно определяемое исходными данными.

Свойства алгоритма

- ***"Понятность"*** для исполнителя –
  - исполнитель алгоритма должен знать, как его выполнить.
- ***"Дискретность"*****(**прерывность, раздельность) –
  - алгоритм должен представлять процесс решения задачи как последовательное выполнение простых (или ранее определённых) шагов (этапов).
- ***"Определённость"*** –
  - каждое правило алгоритма должно быть чётким, однозначным и не оставлять места для произвола.

Выполнение алгоритма носит механический характер и не требует никаких дополнительных указаний или сведений о решаемой задаче.

Свойства алгоритма

- ***"Результативность"***(или конечность)
  - алгоритм должен приводить к решению задачи за конечное число шагов.
- ***"Массовость"*** –
  - алгоритм решения задачи разрабатывается в общем, виде, т. е. он должен быть применим для некоторого класса задач, различающихся лишь исходными данными.
  - (При этом исходные данные могут выбираться из некоторой области, которая называется областью применимости алгоритма).

Особенности алгоритма

- Алгоритм может быть предназначен для выполнения его человеком или автоматическим устройством.
- Ее можно поручить субъекту или объекту, который не обязан вникать в существо дела, а возможно, и не способен его понять. Такой субъект или объект принято называть ***формальным исполнителем***.
- *Каждый алгоритм создается в расчете на вполне конкретного исполнителя.*
- Те действия, которые может совершать исполнитель, называются ***допустимыми действиями***.
- Совокупность допустимых действий образует ***систему команд исполнителя***.
- Алгоритм должен содержать только те действия, которые допустимы для данного исполнителя.
- Объекты, над которыми исполнитель может совершать действия, образуют так называемую ***среду исполнителя***.

Способы записей алгоритмов

- ***Словесно-формульное описание*** (на естественном языке с использованием математических формул).
  - состоит из перечня действий (шагов), каждый из которых имеет порядковый номер.
  - должен выполняться последовательно шаг за шагом.
  - применяют при решении несложных задач.
- ***Графическое описание в виде блок-схемы.***
  - Для обозначения шагов решения в виде схемы алгоритма используются специальные обозначения (символы).
- ***Псевдокоды***
  - полуформализованные описания алгоритмов на условном алгоритмическом языке
- ***Описание на каком-либо языке программирования (программа).***
  - ***Программа*** — это набор машинных команд, который следует выполнить компьютеру для реализации того или иного алгоритма.
  - ***Программа*** — это форма представления алгоритма для исполнения его машиной.

ВГУЭС

Словесно-формульное описание алгоритма. Пример

Задача о сортировке шариков

- Имеются три урны (белая, черная и полосатая). В полосатой урне находятся белые и черные шарики. Надо все черные шарики переложить в черную урну, а белые - в белую. Сортировка производится так: по очереди вынимаются шарики из полосатой урны и в зависимости от цвета кладутся или в черную или в белую урну.

Алгоритм:

1) взять шарик из полосатой урны;
2) если он белый, то опустить в белую урну;
3) если он черный, то опустить в черную урну;
4) если полосатая урна не пуста,

то перейти к действию 1;

1) конец.

Словесно-формульное описание алгоритма

Словесный способ не имеет широкого распространения, так как такие описания:

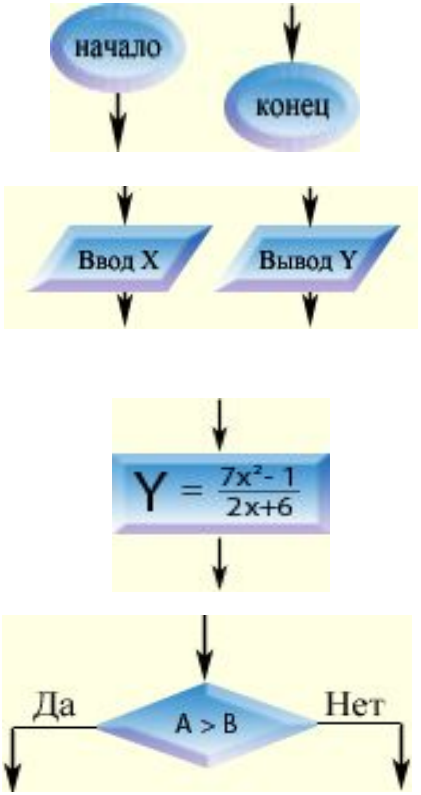
- строго не формализуемы;
- страдают многословностью записей;
- допускают неоднозначность толкования отдельных предписаний.

Графический способ записи алгоритмов

- Является более компактным и наглядным по сравнению со словесным.
- Алгоритм изображается в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий.
- Такое графическое представление называется схемой алгоритма или ***блок-схемой***.
- В блок-схеме каждому типу действий (вводу исходных данных, вычислению значений выражений, проверке условий, управлению повторением действий. окончанию обработки и т. п.) соответствует геометрическая фигура, представленная в виде ***блочного символа***.
- Блочные символы соединяются ***линиями переходов***, определяющими очередность выполнения действий.

Графический способ записи алгоритмов

Наиболее часто употребляемые символы



Псевдокоды

- ***Псевдокоды*** (полуформализованные описания алгоритмов на условном алгоритмическом языке, включающие как элементы языка программирования, так и фразы естественного языка, общепринятые математические обозначения и другое).
- ***Псевдокод*** представляет собой систему обозначений и правил, предназначенную для единообразной записи алгоритмов.
- Псевдокод занимает промежуточное место между естественным и формальными языками.
- В псевдокоде не приняты строгие синтаксические правила для записи команд.
- В псевдокоде есть ***служебные слова***, смысл которых определен раз и навсегда.
- ***Служебные слова*** выделяются в печатном тексте жирным шрифтом, а в рукописном тексте подчеркиваются.

Псевдокоды

Примером псевдокода является *школьный алгоритмический язык*.

Общий вид алгоритма:

алг название алгоритма (аргументы и результаты)
  дано условия применимости алгоритма
  надо цель выполнения алгоритма
нач описание промежуточных величин
    последовательность команд (тело алгоритма)
кон

Программный способ записи

Пример:

Программа нахождения квадрата числа на языке Бейсик

```
10 INPUT "ввести значения x"; x
20 y =x^2
30 PRINT "y ="; y
40 END
```

Виды алгоритма.
Линейный алгоритм

Алгоритм, в котором все этапы решения задачи выполняются строго последовательно.

Например, алгоритм решения математической задачи нахождения гипотенузы, если известны катеты.

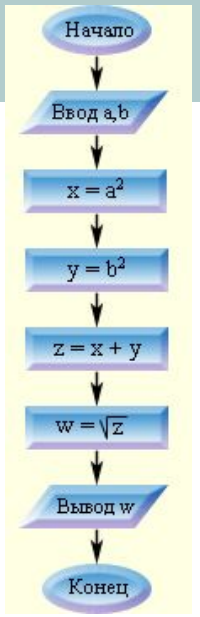
Словесный способ записи:

Возвести первый катет в квадрат;
Возвести второй катет в квадрат;
Сложить результаты действий 1 и 2;
Вычислить квадратный корень из результата 3-го действия и принять его за значение гипотенузы.

Программный способ записи:

```
10 INPUT a,b
20 x=a^2
30 y=b^2
40 z=x+y
50 w=SQR(z)
60 PRINT w
70 END
```

Запись блок-схемой:



ВГУЭС

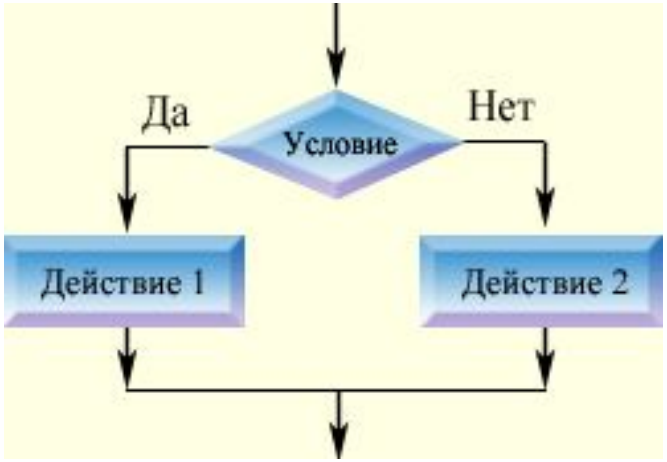
Виды алгоритма.
Разветвляющийся алгоритм

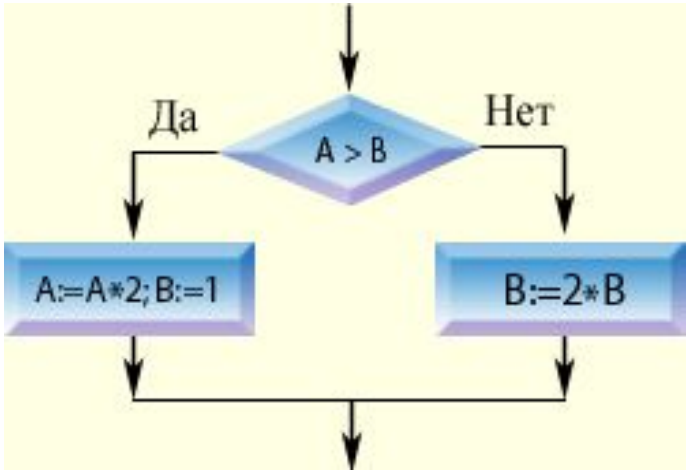
Алгоритм, который выполняется в зависимости от условия, т.е. от вопроса на который можно ответить "да« (истина) или "нет« (ложь).

Полная форма

Это форма записи развлетвляющегося алгоритма, в которой предусмотрены команды в ветви "да" и в ветви "нет".

Если-то-иначе



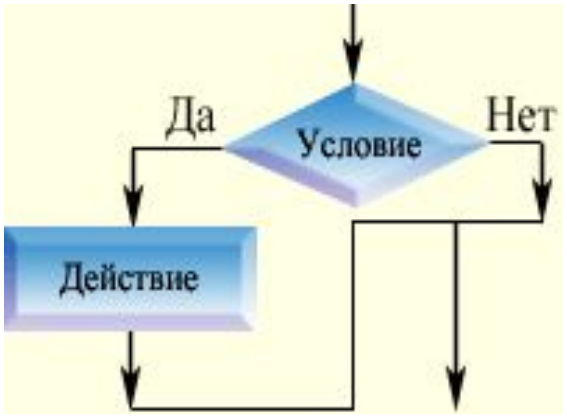


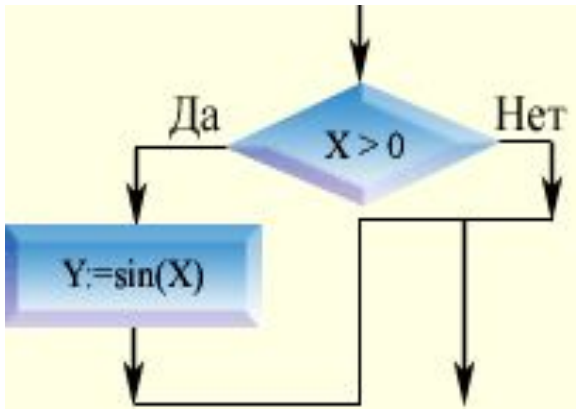
Виды алгоритма.
Разветвляющийся алгоритм

Неполная форма

Это форма записи разветвляющегося алгоритма, в которой предусмотрены команды только в одной ветви.

Если-то-иначе



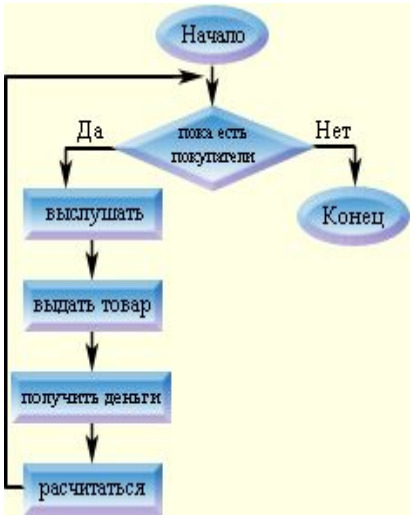


Виды алгоритма.
Циклический алгоритм

Алгоритм, действия которого повторяются.

Существует два типа циклических алгоритмов:

Цикл типа "Пока"



Пример: алгоритм продавца по обслуживанию покупателей

Цикл типа "Для"

Пример: алгоритм учителя по проверке тетрадей учеников

Циклический алгоритм типа "Для"

- Циклический алгоритм типа "Для" - это такой циклический алгоритм, в котором число повторений известно.

  Для организации циклов с известным числом повторений (типа "Для") используют оператор FOR - NEXT.

```
10 FOR A = L TO R STEP N
20 P
30 NEXT A
```

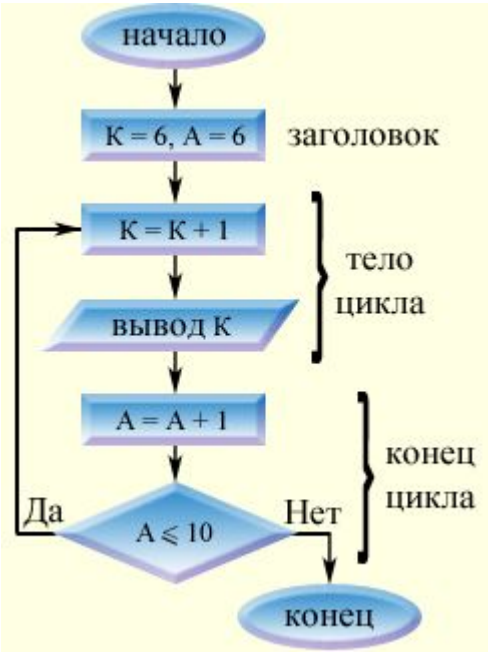
A - счетчик цикла (управляющая переменная)
L - начальное значение (число)
R - коненчное значение (число)
N - шаг цикла (число)
P - тело цикла (последовательность действий)

Циклический алгоритм типа "Для"

Рассмотрим применение оператора FOR - NEXT на следующем примере, где

- L (начальное значение счетчика цикла) = 1,
- R(конечное значение счетчика цикла) = 10:

```
10 K = 6
20 FOR A = 6 TO 10
30 K = K + 1
40 PRINT K;
50 NEXT A
60 END
заголовок - 20
тело цикла - 30; 40
конец цикла - 50
```

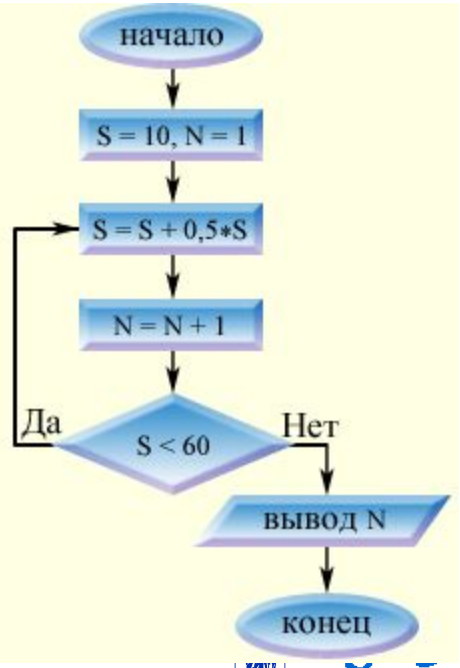


Циклический алгоритм типа "Пока"

- Циклический алгоритм типа "Пока" - это такой циклический алгоритм, действия которого будут выполнятся до тех пор пока выполняется заданное условие.

Начав тренировки, спортсмен в первый день пробежал 10 км. Каждый следующий день он увеличивал дневную норму на 50% от нормы предыдущего дня. Через сколько дней спортсмен пробежит суммарный путь 60 км?

```
10 S = 10
20 N = 1
30 S = S + 0, 5*S
40 N = N + 1
50 IF S<60 THEN
GOTO 30
60 PRINT N
70 END
S - счетчик км
N - счетчик дней
```



Общие положения программирования

- ***Язык программирования низкого уровня***
  - ориентирован на конкретный тип процессора и учитывает его особенности
  - операторы языка близки к машинному коду и ориентированы на конкретные команды процессора.
- ***Языки программирования высокого уровня***
  - *значительно ближе и понятнее человеку, нежели компьютеру.*
  - *создаваемые программы на легко переносимы на другие платформы, для которых создан транслятор этого языка.*
- ***Процесс программирования на универсальном языке высокого уровня состоит из следующих действий:***
  - *ввода и редактирования текста программы,*
  - *трансляции*
  - *отладки.*
- *Процесс поиска ошибок в программе называется **тестированием**,*
- *Процесс устранения ошибок – **отладкой**.*

Базовые конструкции программирования. *Алгоритмическое (модульное) программирование*

- ***Основная идея алгоритмического программирования*** — разбиение программы на последовательность модулей, каждый из которых выполняет одно или несколько действий.
- Алгоритм языке программирования записывается с помощью:
  - команд описания данных;
  - вычисления значений;
  - управления последовательностью выполнения программы.

Структурное программирование

Подпрограммы

- При создании средних по размеру приложений (несколько тысяч строк исходного кода) используется ***структурное программирование,***
  - структура программы должна отражать структуру решаемой задачи, чтобы алгоритм решения был ясно виден из исходного текста.
  - Для этого надо иметь средства для создания программы не только с помощью трех простых операторов, но и с помощью средств, более точно отражающих конкретную структуру алгоритма.
- С этой целью в программирование введено понятие ***подпрограммы*** — набора операторов, выполняющих нужное действие и не зависящих от других частей исходного кода.
  - Программа разбивается на множество мелких подпрограмм, каждая из которых выполняет одно из действий, предусмотренных исходным кодом.
  - Комбинируя эти подпрограммы, удается формировать итоговый алгоритм уже не из простых операторов, а из законченных блоков кода, имеющих определенную смысловую нагрузку, причем обращаться к таким блокам можно по названиям.
  - Получается, что подпрограммы — это новые операторы или операции языка, определяемые программистом.
- Возможность применения подпрограмм относит язык программирования к классу ***процедурных языков***.

Структурное программирование

Процедуры и функции

- Подпрограммы бывают двух видов - ***процедуры и функции.***
- ***Процедура*** просто выполняет группу операторов.
- ***Функция*** вдобавок вычисляет некоторое значение и передает его обратно в главную программу (возвращает значение).
- Чтобы работа подпрограммы имела смысл, ей надо получить данные из внешней программы, которая эту подпрограмму ***вызывает.*** Данные передаются подпрограмме в виде ***параметров*** или ***аргументов,*** которые обычно описываются в ее заголовке так же, как переменные.

Управление последовательностью вызова подпрограмм

- Подпрограммы активизируются ***только*** в момент их вызова.
- Пока выполнение подпрограммы полностью не закончится, оператор главной программы, следующий за командой вызова подпрограммы, выполняться не будет.

Структура подпрограммы

- Подпрограмма состоит из нескольких частей:
  - заголовка с параметрами,
  - тела подпрограммы (операторов, которые будут выполняться при ее вызове)
  - завершения подпрограммы.
- Локальные переменные, объявленные внутри подпрограммы, имеют областью действия только ее тело.
- После того как функция рассчитала нужное значение, ей требуется явно вернуть его в вызывающую программу. Для этого может использоваться специальный оператор или особая форма оператора присваивания, когда в левой части указывается имя функции, а справа — возвращаемое значение.

Объектно-ориентированное программирование

Понятие объекта

- Реальные объекты окружающего мира обладают тремя базовыми характеристиками.
- Они имеют набор ***свойств***, способны разными ***методами*** изменять эти свойства, реагировать на ***события***, возникающие как в окружающем мире, так и внутри объекта.
- Именно в таком виде в языках программирования и реализовано понятие ***объекта,*** как совокупности ***свойств*** (***структур данных***, характерных для этого объекта), ***методов*** их обработки (подпрограмм изменения свойств) и ***событий.***
- Появление возможности создания объектов в программах качественно улучшили производительность труда программистов.

Объектно-ориентированное программирование

Класс – новый тип данных

- Объекты могут иметь идентичную структуру и отличаться только типом свойств.
- В программе создается новый тип, основанный на структуре объекта. Он называется ***классом,*** а каждый конкретный объект, имеет структуру этого класса, и называется ***экземпляром класса.***

Объектно-ориентированное программирование

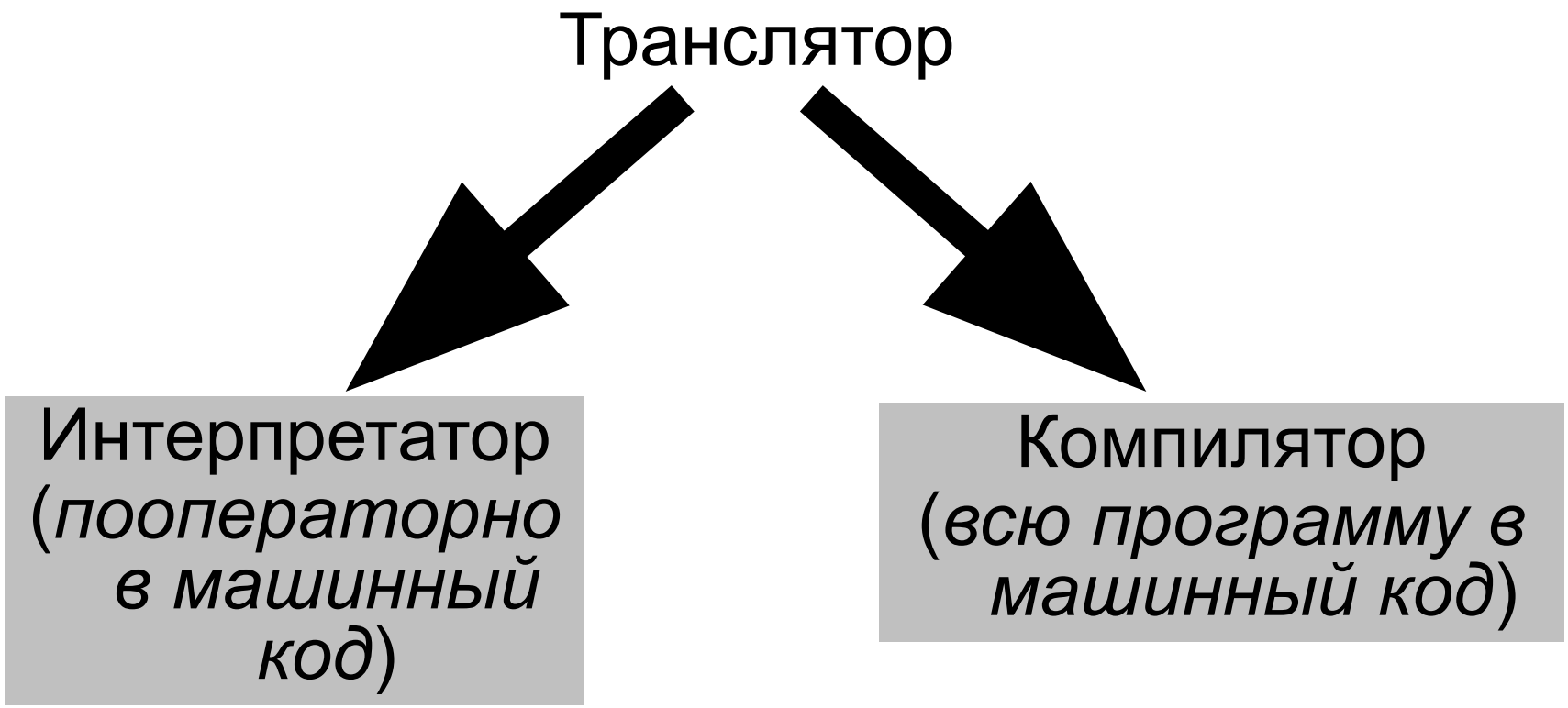
Описание нового класса

- Описание нового класса похоже на описание новой структуры данных: к полям (свойствам) добавляются методы — подпрограммы.
  - В Си++ и Паскале для описания класса используется ключевое слово.
- При определении подпрограмм, принадлежащих конкретному классу, его методов, заголовке подпрограммы перед ее названием явно указывается, к какому классу она принадлежит.
  - Название класса от названия метода отделяют специальные символы (точка в Паскале или два двоеточия в Си++).
- Доступ к свойствам объектов и к их методам осуществляется так же, как к полям записей, через точку.
- Объединение данных с методами в одном типе (классе) ***называется инкапсуляцией***. Важнейшая характеристика класса — возможность создания на его основе новых классов с ***наследованием*** всех его свойств и методов и добавлением собственных. Класс, не имеющий предшественника, называется ***базовым***.

Построение программ

- Компьютерные программы создают ***программисты*** — люди, обученные процессу составления программ *(**программированию**).*
- Программирование сводится к созданию последовательности команд, необходимой для решения определенной задачи.
- Для представления алгоритма в виде, понятном компьютеру, служат ***языки программирования**.*
- Сначала всегда разрабатывается ***алгоритм*** действий, а потом он записывается на одном из таких языков.
- В итоге получается текст ***программы*** - законченное и детальное описание алгоритма на ***языке программирования.***
- Затем этот текст программы специальными служебными приложениями, которые называются ***трансляторами**,* либо переводится в машинный код, либо исполняется.

Компиляторы и интерпретаторы



ВГУЭС

Компиляторы и интерпретаторы

- С помощью языка программирования создается не готовая программа, а только ее текст, описывающий ранее разработанный алгоритм.
- Чтобы получить работающую программу, надо ее 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***).
- **Интерпретатор** берет очередной оператор языка из текста программы, анализирует его структуру и затем сразу исполняет (обычно после анализа оператор транслируется в некоторое промежуточное представление или даже машинный код для более эффективного дальнейшего исполнения).
- **Компиляторы** полностью обрабатывают весь текст программы (он иногда называется *исходный код).*
  - Они просматривают его в поисках синтаксических ошибок (иногда несколько раз), выполняют определенный смысловой анализ и затем автоматически переводят ***(транслируют)*** на машинный язык — генерируют машинный код.
  - В результате законченная программа получается компактной и эффективной, работает в сотни раз быстрее программы, выполняемой с помощью интерпретатора, и может быть перенесена на другие компьютеры с процессором, поддерживающим соответствующий машинный код.
- В реальных системах программирования перемешаны технологии и компиляции и интерпретации.
  - В процессе отладки программа может выполняться по шагам, а результирующий код не обязательно будет машинным.

Системы программирования

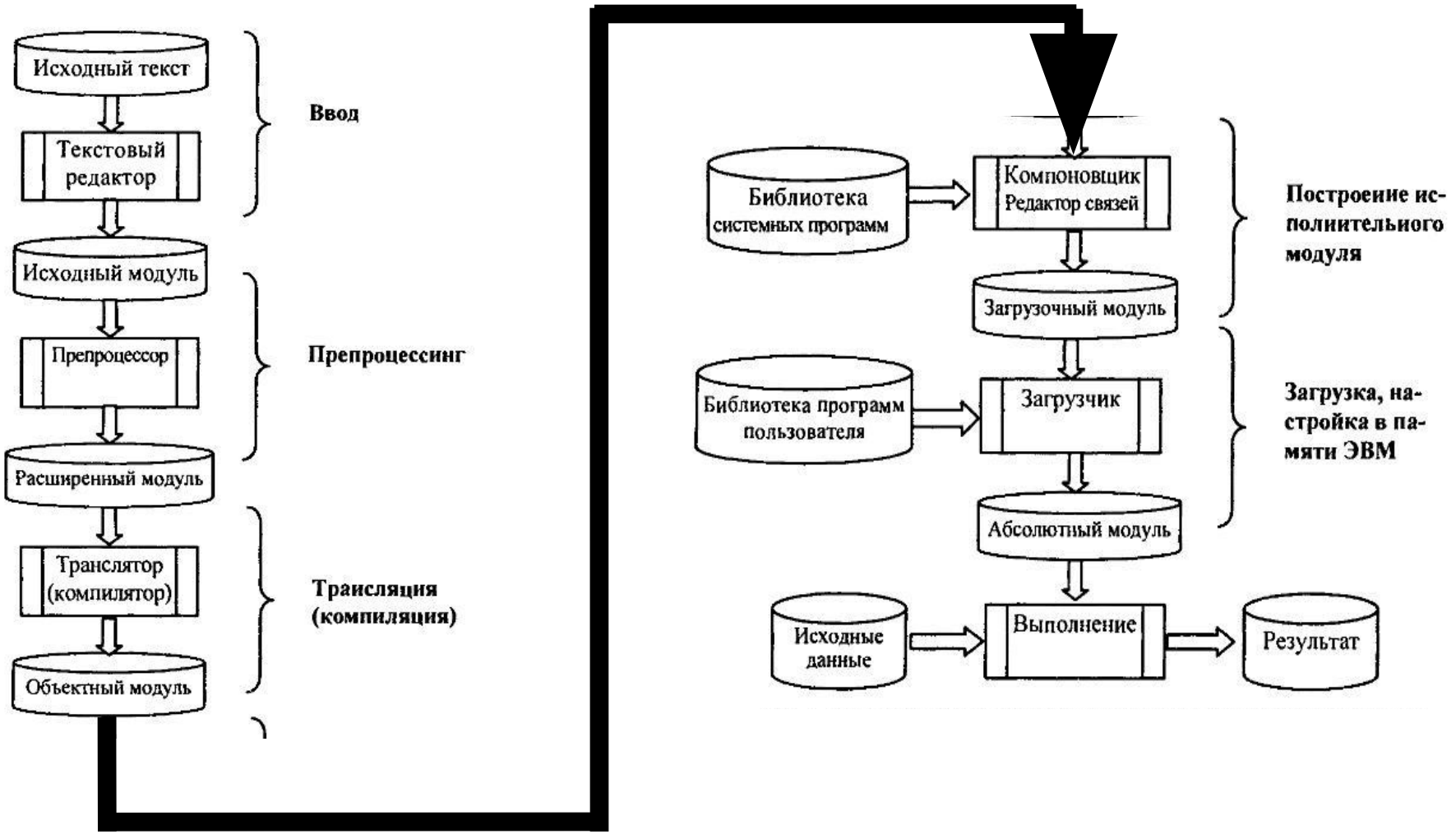
В общем случае для создания программы на языке программирования нужно иметь следующие компоненты:

1. ***Текстовый редактор***.
   - формировать текст программы можно в любом редакторе, получая в итоге текстовый файл с исходным *текстом* программы.
   - Лучше использовать специализированные редакторы, ориентированные на конкретный язык программирования и позволяют в процессе ввода автоматически проверять правильность программы.
2. Исходный текст с помощью ***программы-компилятора*** переводится в машинный код.
   - компилятор обычно выдает промежуточный объектный код (двоичный файл, стандартное расширение .OBJ).

Системы программирования

3. Исходный текст большой программы состоит, как правило, из нескольких ***модулей*** (файлов с исходными текстами),
   - Каждый модуль компилируется в отдельный файл с объектным кодом, которые затем надо объединить в одно целое.
4. ***Исполнимый код*** – 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.

Системы программирования



Интегрированные системы программирования

Для создания программы нужны:

- текстовый редактор
- компилятор,
- редактор связей,
- библиотеки функций.
- 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.

ВГУЭС

Основные системы программирования

Из универсальных языков программирования сегодня наиболее популярны:

- **Бейсик (Basic)** — для освоения требует начальной подготовки (общеобразовательная школа);
- **Паскаль (Pascal)** — требует специальной подготовки;
- **Си++ (C++), Ява (Java)** — требуют профессиональной подготовки.

Для каждого из этих языков программирования сегодня имеется немало систем программирования, выпускаемых различными фирмами и ориентированных на различные модели ПК и операционные системы.

Основные системы программирования

Наиболее популярны следующие визуальные среды быстрого проектирования программ для Windows:

- Basic: **Microsoft Visual Basic**
- Pascal: **Borland Delphi**
- C++: **Borland C++ Bulider**
- Java: **Symantec Cafe, NetBeans(для UNIX)**

Для разработки серверных и распределенных приложений можно использовать программные продукты многих фирм, например, систему программирования Microsoft Visual C++.

Архитектура программных систем

- крупные автоматизированные комплексы (например, система автоматизации предприятия) состоят из десятков и сотен отдельных программ, которые взаимодействуют друг с другом по сети, выполняясь на различных компьютерах.
- В таких случаях говорят, что работают в различной программной архитектуре.

Архитектура программных систем

- *Автономные приложения.*
- *Приложения в файл-серверной архитектуре.*
- *Приложения в клиент-серверной архитектуре.*
- *Приложения в многозвенной архитектуре.*
- *Приложения в распределенной архитектуре.*

Специальные технологии, позволяющие создавать программу в виде набора компонентов, которые можно запускать на любых серверах, связанных в сеть (компоненты как бы распределны по сети).
Преимущество - при выходе из строя любого компьютера специальные программы-мониторы, которые следят за корректностью работы компонентов.
Надежность системы очень высокая, а вычислительная загрузка разделяется между серверами оптимальным образом.
Доступ к возможностям любого компонента осуществляется с произвольного клиентского места. Так как все вычисления происходят на серверах, появляется возможность создавать сверхтонкие клиенты – программы, только отображающие получаемую из сети информацию и требующие минимальных компьютерных ресурсов.
Доступ к компонентной системе возможет не только с ПК, но и с небольших мобильных устройств.

Виды программирования

- **Структурное программирование**
  - структура программы должна отражать структуру решаемой задачи, чтобы алгоритм решения был ясно виден из исходного текста.
- **Объектно-ориентированное программирование**
  - *объект,* как совокупности *свойств* (структур данных, характерных для этого объекта), *методов* их обработки (подпрограмм изменения свойств) и *событий,* на данный объект может реагировать и которые приводят, как правило, к изменению свойств объекта.

Виды программирования

- **Событийно-ориентированное программирование**
  - Идеология системы Windows основана на событиях. Щелкнул человек на кнопке, выбрал пункт меню, нажал на клавишу или кнопку мыши — в Windows генерируется подходящее *сообщение,* которое отсылается окну соответствующей программы.
  - Структура программы, созданной с помощью событийного программирования.
    - Главная часть представляет собой один бесконечный цикл, который опрашивает Windows, следя за тем, не появилось ли новое сообщение.
    - При его обнаружении вызывается подпрограмма, ответственная за *обработку* соответствующего события (обрабатываются только нужные события), и подобный цикл опроса продолжается, пока не будет получено сообщение «Завершить работу».
    - События могут быть *пользовательскими,* возникшими в результате действий пользователя, *системными,* возникающими в операционной системе (например, сообщениями от таймера), и *программными,* генерируемыми самой программой (например, обнаружена ошибка, и ее надо обработать).

Языки программирования Основные понятия. Алфавит. Синтаксис. Семантика

Алгоритмический язык (как и любой другой язык), образуют три его составляющие: **алфавит, синтаксис и семантика**.

- ***Алфавит*** – фиксированный для данного языка набор символов (букв, цифр, специальных знаков и т.д.), которые могут быть использованы при написании программы.
- ***Синтаксис*** - правила построения из символов алфавита специальных конструкций, с помощью которых составляется алгоритм.
- ***Семантика*** - система правил толкования конструкций языка. Таким образом, программа составляется с помощью соединения символов алфавита в соответствии с синтаксическими правилами и с учетом правил семантики.

Основные элементы алгоритмического языка

- **Имена (идентификаторы)** - последовательность символов для обозначения объектов программы (переменных, массивов, функций и др.).
- **Операции**. Существуют следующие типы операций:
  - - **арифметические операции**: сложение, обозначается символом “+”; вычитание, обозначается символом “-”; умножение, обозначается символом “*”; деление, обозначается символом “/” и др. ;
  - - **логические операции**: операции “логическое и”, “логическое или”, “логическое не” и др.;
  - - **операции отношения**: меньше, обозначается символом “<”; больше, обозначается символом “>”; меньше или равно, обозначается символами “<=”; больше или равно, обозначается символами “>=”; равно, обозначается символом “=”; не равно, обозначается символами “<>”.
  - - **операция конкатенации** символьных значений друг с другом, изображается знаком "+".

ВГУЭС

Основные элементы алгоритмического языка

- **Ключевые слова** – это слова языка, имеющие строго определенное назначение, которые не могут использоваться в качестве идентификаторов.
- **Данные** - величины, обрабатываемые программой. Имеется три основных вида данных:
  - **константы**,
  - **переменные**
  - и **массивы**.
- **Выражения** – элементы языка, которые предназначаются для выполнения необходимых вычислений, состоят из констант, переменных, указателей функций, объединенных знаками операций. Выражения записываются в виде линейных последовательностей символов.

Виды данных

- **Константы** - это данные, которые зафиксированы в тексте программы и не изменяются в процессе ее выполнения.

  Примеры констант:

  - **числовые**: 7.5, 12;
  - **логические**: true(истина), false(ложь);
  - **символьные**: "A", "+";
  - **строковые**: "abcde", "информатика".
- **Переменные** – это данные, которые могут изменять свои значения в ходе выполнения программы. Они обозначаются именами. Переменные бывают **целые**, **вещественные**, **логические**, **символьные** и **строковые**.
- **Массивы** - последовательности однотипных элементов, число которых фиксировано и которым присвоено одно имя. Положение элемента в массиве однозначно определяется его индексами - одним в случае одномерного массива, или несколькими, если массив многомерный.

Языки программирования. Основные понятия

- **Оператор** – это элемент языка, который задает полное описание некоторого действия, которое необходимо выполнить. Оператор - это наиболее крупное и содержательное понятие языка: каждый оператор представляет собой законченную фразу языка программирования и определяет некоторый вполне законченный этап обработки данных. В состав операторов входят ключевые слова; данные; выражения и т.д.
- **Стандартная функция** – подпрограмма, заранее встроенная в транслятор языка для вычисления часто употребляемых функций. В качестве аргументов функций можно использовать константы, переменные и выражения.
- **Программа** - это последовательность инструкций, предназначенных для выполнения компьютером. В настоящее время программы оформляются в виде текста, который записывается в файлы.

Языки программирования. Основные понятия

- Языки высокого уровня работают через трансляционные программы -**трансляторы**, которые преобразуют исходный код в последовательность команд машинного языка.
- Существует два основных вида трансляторов:
  - **интерпретаторы**, которые сканируют и проверяют исходный код в один шаг,
  - и **компиляторы**, которые сканируют исходный код для создания текста программы на машинном языке, которая затем выполняется отдельно.

Языки программирования. Основные понятия

- В общем случае программа может иметь **модульную структуру**, т.е. состоять из нескольких программных единиц, связанных между собой командами передачи управления.
- Такой принцип построения программ называется **модульным**.
- Программная единица, с первой команды которой начинается выполнение программы, называется **головной программой**.
- Остальные программные единицы, входящие в единую программу, называются **подпрограммами**.
- **Подпрограмма** - это последовательность операторов, которые определены и записаны только в одном месте программы, однако их можно вызвать для выполнения из одной или нескольких точек программы.

Языки программирования. Основные понятия

- **Функция** - это программная единица, которая может быть употреблена в выражении. Функция прямо возвращает величину, которая используется при вычислении этого выражения, и, кроме того, может возвращать величины через параметры.
- **Подпрограммы и функции** позволяют создавать большие структурированные программы, которые можно делить на части.

Это дает преимущества в следующих ситуациях:

1. Если программа большая, разделение ее на части облегчает создание, тестирование и ее сборку.
2. Если программа большая и повторная компиляция всего исходного текста занимает много времени, разделение ее на части экономит время компиляции.
3. Если процедуру надо использовать в разных случаях разным образом, можно записать ее в отдельный файл и скомпилировать отдельно.

Использование материалов презентации

Использование данной презентации, может осуществляться только при условии соблюдения требований законов РФ об авторском праве и интеллектуальной собственности, а также с учетом требований настоящего Заявления.

Презентация является собственностью авторов. Разрешается распечатывать копию любой части презентации для личного некоммерческого использования, однако не допускается распечатывать какую-либо часть презентации с любой иной целью или по каким-либо причинам вносить изменения в любую часть презентации. Использование любой части презентации в другом произведении, как в печатной, электронной, так и иной форме, а также использование любой части презентации в другой презентации посредством ссылки или иным образом допускается только после получения письменного согласия авторов.