Алгоритмізація і програмування

Пецков Р.О.
викл.каф.АКІТТ

Лекція №1,2

Етапи вирішення задач

Для успішного використання ПК у своїй професійній діяльності користувач повинен вміти формулювати завдання, розробляти алгоритми їх вирішення, записувати алгоритми мовою, зрозумілою ПК.

Етапи підготовки і вирішення реальних завдань включають:

- постановку задачі
- фізичне моделювання
- математичне або інформаційне моделювання
- алгоритмізацію завдання
- розробку програми
- тестування і налагодження програм
- аналіз результатів

Алгоритми та способи їх описання

- Алгоритм - точний порядок дій, що визначає процес, що веде від вихідних даних до шуканого результату.
- Алгоритм - це кінцева послідовність однозначних розпоряджень, виконання яких дозволяє за допомогою кінцевого числа кроків отримати рішення задачі, однозначно обумовлений вихідними даними.

Властивості алгоритму

"Зрозумілість" для виконавця -

- виконавець алгоритму повинен знати, як його виконати.

"Дискретність" (переривчастість, роздільність) -

- алгоритм повинен представляти процес вирішення завдання як послідовне виконання простих (або раніше визначених) кроків (етапів).

"Визначеність" -

- кожне правило алгоритму має бути чітким, однозначним і не залишати місця для свавілля.

Виконання алгоритму носить механічний характер і не вимагає ніяких додаткових вказівок, або відомостей про задачу, що вирішується.

Властивості алгоритму

"Результативність" (або кінцівку)

- алгоритм повинен призводити до вирішення завдання за кінцеве число кроків.

"Масовість" -

- алгоритм вирішення задачі розробляється в загальному, вигляді, тобто він повинен бути застосовний для деякого класу задач, що розрізняються лише вихідними даними.
- (При цьому вихідні дані можуть вибиратися з деякої області, яка називається областю застосовності алгоритму).

Особливості алгоритму

- Алгоритм може бути призначений для виконання його людиною або автоматичним пристроєм.
- Його можна доручити суб'єкту чи об'єкту, який не зобов'язаний вникати в суть справи, а можливо, і не здатний його зрозуміти. Такий суб'єкт або об'єкт прийнято називати формальним виконавцем.
- Кожен алгоритм створюється з розрахунку на цілком конкретного виконавця.
- Ті дії, які може вчиняти виконавець, називаються допустимими діями.
- Сукупність допустимих дій утворює систему команд виконавця.
- Алгоритм повинен містити тільки ті дії, які допустимі для даного виконавця.
- Об'єкти, над якими виконавець може вчиняти дії, утворюють так зване середовище виконавця.

Способи запису алгоритмів

Словесно-формульне опис (природною мовою з використанням математичних формул).

- складається з переліку дій (кроків), кожен з яких має порядковий номер.
- повинен виконуватися послідовно крок за кроком.
- застосовують при вирішенні нескладних завдань.

Графічний опис у вигляді блок-схеми.

- Для позначення кроків вирішення у вигляді схеми алгоритму використовуються спеціальні позначення (символи).

Псевдокоди

- напівформалізоване опису алгоритмів на умовному алгоритмічній мові

Опис на мові програмування (програма).

- Програма - це набір машинних команд, який слід виконати комп'ютеру для реалізації того чи іншого алгоритму.
- Програма - це форма подання алгоритму для виконання його машиною.

Словесно-формульний опис алгоритму. Приклад

Завдання: сортування кульок

Є три урни (біла, чорна і смугаста). В смугастій урні знаходяться білі і чорні кульки. Треба всі чорні кульки перекласти в чорну урну, а білі - в білу. Сортування проводиться так: по черзі виймаються кульки зі смугастої урни і в залежності від кольору кладуться або в чорну або в білу урну.

Алгоритм:

1. взяти кульку зі смугастої урни;
2. якщо він білий, то опустити в білу урну;
3. якщо він чорний, то опустити в чорну урну;
4. якщо смугаста урна не порожня,то перейти до дії 1;
0. кінець.

Словесно-формульний опис алгоритму

Словесний спосіб не має широкого розповсюдження, оскільки такі описи:

- строго не формалізуються;
- страждають багатослівністю записів;
- допускають неоднозначність тлумачення окремих приписів.

Графічний спосіб запису алгоритмів

- Являється більш компактним і наочним порівняно зі словесним.
- Алгоритм зображується у вигляді послідовності пов'язаних між собою функціональних блоків, кожен з яких відповідає виконанню одного або кількох дій.
- Таке графічне представлення називається схемою алгоритму або блок-схемою.
- У блок-схемі кожному типу дій (введення вхідних даних, обчисленню значень виразів, перевірці умов, управлінню повторенням дій. Закінченню обробки і т. д.) відповідає геометрична фігура, представлена у вигляді блочного символу.
- Блокові символи з'єднуються лініями переходів, що визначають черговість виконання дій.

Псевдокоди

- Псевдокоди (напівформалізований опис алгоритмів на умовній алгоритмічній мові, що включає в себе як елементи мови програмування, так і фрази природної мови, загальноприйняті математичні позначення та інше).

- Псевдокод являє собою систему позначень і правил, призначену для однакової запису алгоритмів.

- Псевдокод займає проміжне місце між природним і формальними мовами.

- У псевдокоді не прийняті строгі синтаксичні правила для записи команд.

- У псевдокоді є службові слова, зміст яких визначено раз і назавжди.

📫Службові слова виділяються в друкованому тексті жирним шрифтом, а в рукописному тексті підкреслюються.

Псевдокоди

Прикладом псевдокоду є шкільна алгоритмічна мова.

Загальний вид алгоритму:

```
поч назва алгоритму (аргументи і результати)?
    дано умови застосовності алгоритму?
    ціль мета виконання алгоритму
поч
    опис проміжних величин
    послідовність команд (тіло алгоритму)
кін
```

Програмний спосіб запису

Приклад:

Програма знаходження квадрата числа на мові Бейсік:

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

Види алгоритмів. Лінійний алгоритм

Алгоритм, в якому всі етапи рішення завдання виконуються строго послідовно.

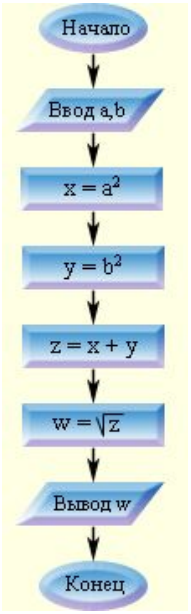
Наприклад, алгоритм розв'язання математичної задачі знаходження гіпотенузи, якщо відомі катети.

Словесний спосіб запису:
Піднести перший катет до квадрату;
Піднести другий катет до квадрату;
Додати результати дій 1 і 2;
Обчислити квадратний корінь з результату 3 дії і прийняти його за значення гіпотенузи.

Программний спосіб запису:

```
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
```

Блок-схема:

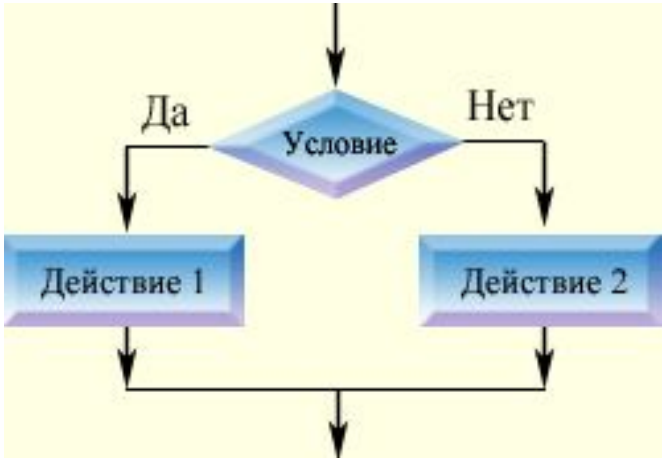


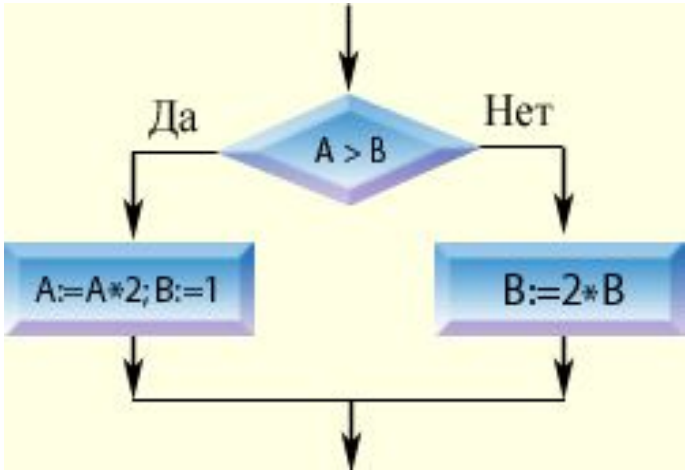
Види алгоритмів. Розгалужений алгоритм

Алгоритм, який виконується в залежності від умови, тобто від питання на яке можна відповісти "так" (істина) або "ні" (неправда).

Повна форма

Це форма запису розгалуженого алгоритму, в якій передбачені команди в гілці "так" і в гілці "ні ".

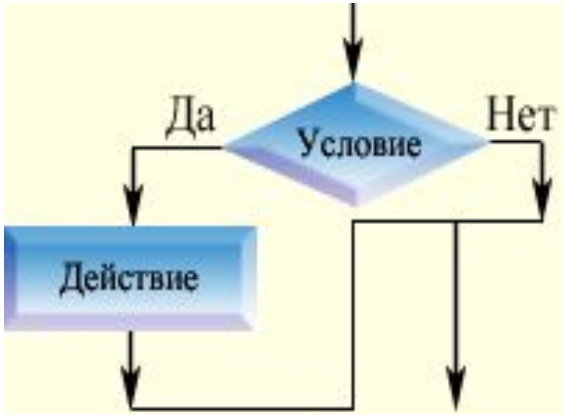


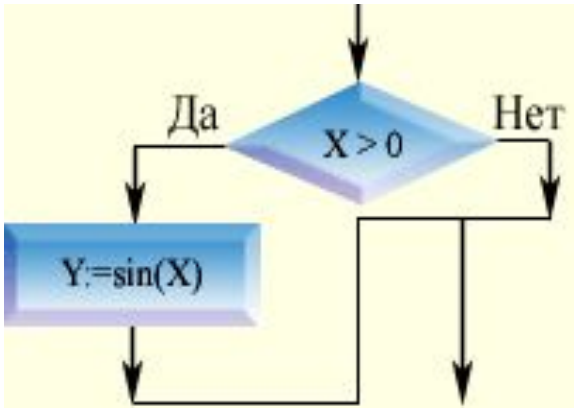


Види алгоритмів. Розгалужений алгоритм

неповна форма

Це форма запису розгалуженого алгоритму, в якій передбачені команди тільки в одній гілці.



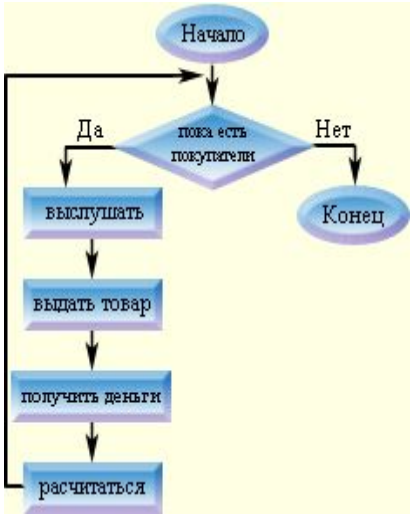


Види алгоритмів. Циклічний алгоритм

Алгоритм, дії якого повторюються.

Існує два типи циклічних алгоритмів:

З передумовою та після умовою



Лекція №3

Побудова програм

Комп'ютерні програми створюють програмісти - люди, навчені процесу складання програм (програмування).

Програмування зводиться до створення послідовності команд, необхідної для вирішення певної задачі.

Для представлення алгоритму у вигляді, зрозумілому комп'ютеру, служать мови програмування.

Спочатку завжди розробляється алгоритм дій, а потім він записується на одному з таких мов.

У підсумку виходить текст програми - закінчене і детальний опис алгоритму на мові програмування.

Потім цей текст програми спеціальними службовими програмами, які називаються трансляторами, або переводиться в машинний код, або виконується.

Загальні положення програмування

Мова програмування низького рівня

-орієнтована на конкретний тип процесора і враховує його особливості

-оператори мови близькі до машинного коду і орієнтовані на конкретні команди процесора.

Мови програмування високого рівня

-значно ближче і зрозуміліше людині, ніж комп'ютеру.

-створювані програми легко переносяться на інші платформи, для яких створено транслятор цієї мови.

Процес програмування на універсальній мові високого рівня складається з наступних дій:

введення і редагування тексту програми,трансляції,налагодження.

Процес пошуку помилок у програмі називається тестуванням,

Процес усунення помилок - налагодженням.

Основні парадигми програмування

Парадигма програмування — це система ідей і понять, які визначають стиль написання комп’ ютерних програм , а також спосібмислення програміста.

-Процедурне програмування
-Об'єктно-орієнтоване програмування
-Функціональне програмування
-Імперативне програмування
-Декларативне програмування
-Прототипне програмування
-Аспектно-орієнтоване програмування
-Предметно-орієнтоване програмування
-Функціонально-орієнтоване програмування
-Структурне програмування
-Модульне програмування
-Збірне програмування
-Програмування з абстрактними типами даних
-Схемне програмування
-Логічне програмування
-Паралельне програмування
-Компонентне програмування
-Агентно-орієнтоване програмування
-Алгебраїчне програмування

Базові конструкції програмування. Алгоритмічне (модульне) програмування

Основна ідея алгоритмічного програмування - розбиття програми на послідовність модулів, кожен з яких виконує одне або декілька дій.

Алгоритм мови програмування записується за допомогою:

команд опису даних;

обчислення значень;

управління послідовністю виконання програми.

Структурне програмування

підпрограми

*При створенні середніх за розміром додатків (кілька тисяч рядків вихідного коду) використовується структурне програмування,

- структура програми повинна відображати структуру розв'язуваної задачі, щоб алгоритм вирішення був ясно видно з вихідного тексту.

- Для цього треба мати засоби для створення програми не тільки за допомогою трьох простих операторів, але і за допомогою засобів, що більш точно відображають конкретну структуру алгоритму.

*З цією метою в програмування введено поняття підпрограми - набору операторів, що виконують потрібну дію і не залежать від інших частин вихідного коду.

Програма розбивається на безліч дрібних підпрограм, кожна з яких виконує одну з дій, передбачених вихідним кодом.

Комбінуючи ці підпрограми, вдається формувати підсумковий алгоритм вже не з простих операторів, а із закінчених блоків коду, що мають певне смислове навантаження, причому звертатися до таких блоків можна за назвами.

Виходить, що підпрограми - це нові оператори або операції мови, що визначаються програмістом.

*Можливість застосування підпрограм відносить мову програмування до класу процедурних мов.

Структурне програмування

Процедури і функції

Підпрограми бувають двох видів - процедури і функції.

Процедура просто виконує групу операторів.

Функція вдобавок обчислює деяке значення і передає його назад в головну програму (повертає значення).

Щоб робота підпрограми мала сенс, їй треба отримати дані з зовнішньої програми, яка цю підпрограму викликає. Дані передаються підпрограмі у вигляді параметрів або аргументів, які зазвичай описуються в її заголовку так само, як змінні.

Управління послідовністю виклику підпрограм

Підпрограми активізуються тільки в момент їх виклику.

Поки виконання підпрограми повністю не закінчиться, оператор головної програми, наступний за командою виклику підпрограми, виконуватися не буде.

Структура підпрограми

Підпрограма складається з декількох частин:

- заголовка з параметрами,
- тіла підпрограми (операторів, які будуть виконуватися при її виклику)
- завершення підпрограми.

Локальні змінні, оголошені в середині підпрограми.

Після того як функція розрахувала потрібне значення, їй потрібно явно повернути його в програму, що викликала. Для цього може використовуватися спеціальний оператор або особлива форма оператора присвоювання.

Лекція №4

Об'єктно-орієнтоване програмування

Об’єктно-орієнтований метод програмування визначає стратегію побудови об’єктної системи, згідно з якою розробники системи мають мислити в термінах об'єктів, а не функцій.

Об'єкт – це певна сутність, що перебуває в різних станах і має певний набір операцій. Операції, що пов’язані з об’єктом, надають іншим об’єктам послуги (сервіси) для виконання певних функцій, а їх стан залежить від значень атрибутів.

Об'єкти створюються відповідно до визначення класу об’ єктів, в якому описуються всі їхні атрибути й операції.

Об'єктно-орієнтоване програмування

поняття об'єкта

Реальні об'єкти навколишнього світу володіють трьома базовими характеристиками.

Вони мають набір властивостей, здатні різними методами змінювати ці властивості, реагувати на події, що виникають як у навколишньому світі, так і всередині об'єкта.

Саме в такому вигляді в мовах програмування і реалізовано поняття об'єкта, як сукупності властивостей (структур даних, характерних для цього об'єкта), методів їх обробки (підпрограм зміни властивостей) і подій.

Поява можливості створення об'єктів в програмах якісно поліпшили продуктивність праці програмістів.

Об'єктно-орієнтоване програмування

Модель об’єктно-орієнтованої програмної системи можна розглядати як набір взаємодіючих об'єктів, що мають власний стан і набір операцій, які впливають на стан інших об'єктів. Об'єкти приховують інформацію про значення станів, операцій і обмежують доступ до них.

Об'єктно-орієнтоване програмування

Клас - новий тип даних

- Об'єкти можуть мати ідентичну структуру і відрізнятися тільки типом властивостей.
- У програмі створюється новий тип, заснований на структурі об'єкта. Він називається класом, а кожен конкретний об'єкт, має структуру цього класу, і називається екземпляром класу.

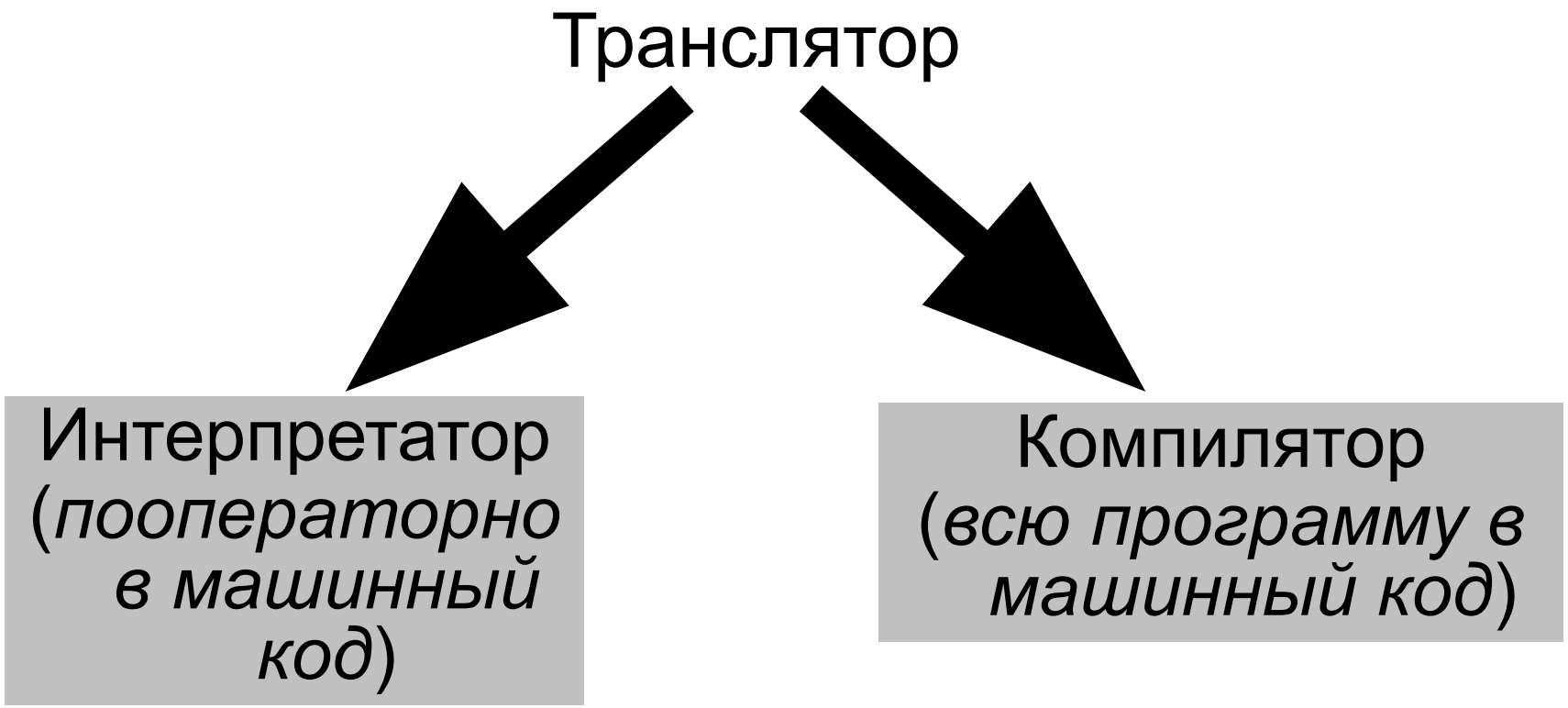
Об'єктно-орієнтоване програмування

Опис нового класу

- Опис нового класу схожий на опис нової структури даних:
до полів (властивостям) додаються методи - підпрограми.
У Сі ++ і JAVA для опису класу використовується ключове слово.
- При визначенні підпрограм, що належать конкретному класу, його методів, заголовку підпрограми перед її назвою явно вказується, до якого класу вона належить.
- Назва класу від назви методу відокремлюють спеціальні символи (крапка в JAVA або двокрапки в Сі ++).
- Доступ до властивостей об'єктів і до їхніх методів здійснюється так само, як до полів записів, через крапку.
- Об'єднання даних з методами в одному типі (класі) називається інкапсуляцією.

Найважливіша характеристика класу - можливість створення на його основі нових класів з успадкуванням всіх його властивостей і методів і додаванням власних. Клас, який не має попередника, називається базовим.

Компиляторы и интерпретаторы



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

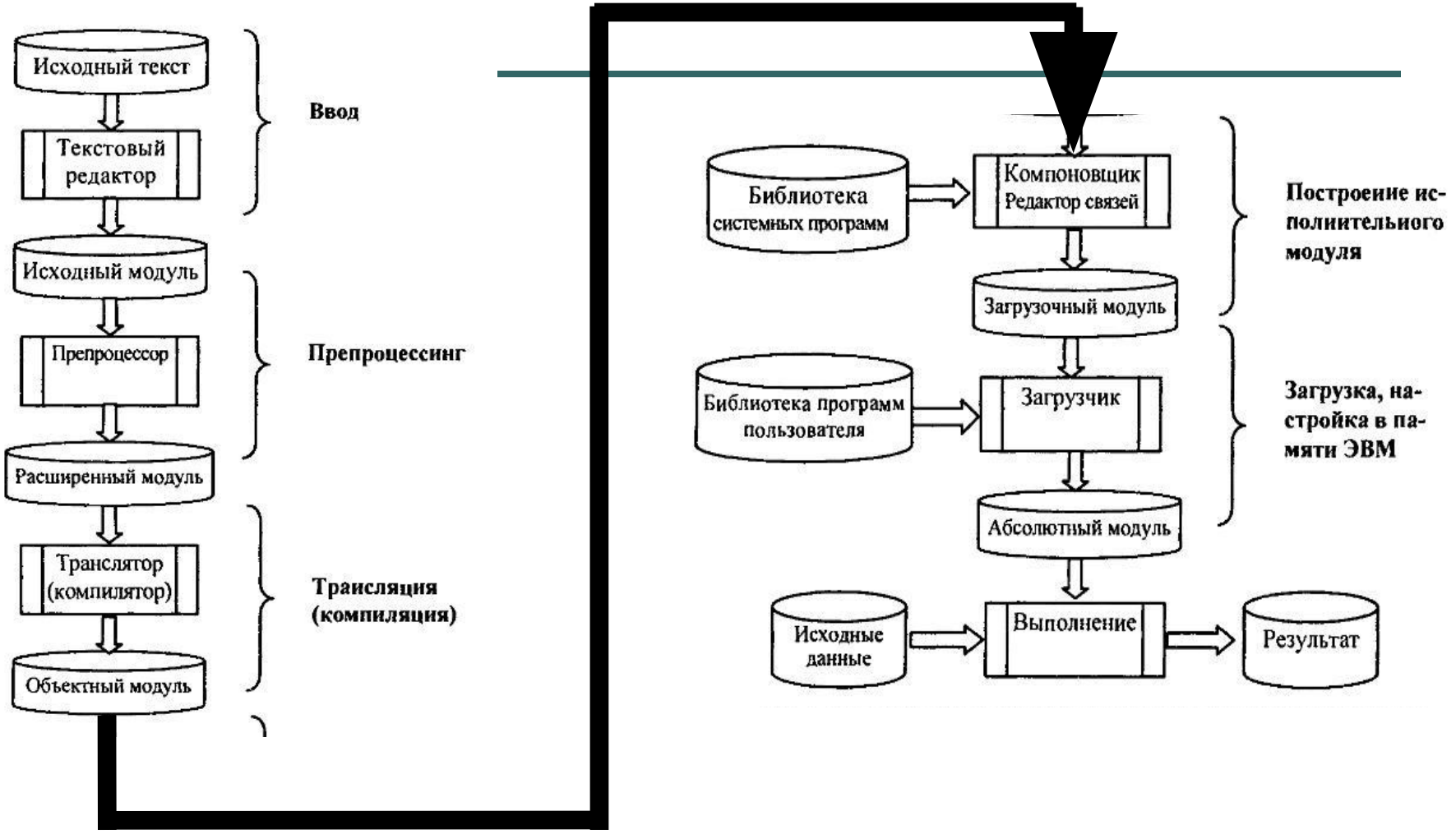
В общем случае для создания программы на языке программирования нужно иметь следующие компоненты:

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.

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.

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 "+"

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

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.