Основы программирования

Введение: алгоритмы и программы

Уравнение: $y = \sin ax + b/2$

Исполнитель-человек:

- **знает**, как решать уравнения, в каком порядке выполняются арифметические операции
- **уточняет**, какие величины заданы и что необходимо вычислить
- **получает (запоминает)** заданные значения
- **использует** таблицы или калькулятор для получения значения синуса (или арксинуса)
- **формирует и выполняет** требуемую последовательность операций

Простейшая модель компьютера

2 основных блока: **память** и **процессор**

Память - последовательность ячеек (байт)

Каждый байт:

- имеет свой номер (адрес)
- может хранить любое целое число в диапазоне 0…255 во внутреннем (двоичном) формате

Возможен **произвольный доступ** к любому байту памяти

Время доступа не зависит от адреса байта

Принцип программного управления

- Работой компьютера управляет программа, которая задает совокупность команд и порядок их выполнения
- Программа и данные, необходимые для ее работы, хранятся в памяти
- Процессор читает из памяти и выполняет команды в нужном порядке

Команды компьютера

Каждая команда имеет фиксированную длину в байтах и фиксированный формат

Пример с символическими кодами команд (Ассемблер):

```
mov ax,[$0064cba0]
add ax,[$0064cba4]
mov [$0064cba8],ax
```

Языки программирования

Фиксированная структура программы определяет совокупность действий и порядок их выполнения

Операторы позволяют описать необходимые действия (команды) на языке, понятном человеку

Переменные различных типов позволяют создавать, хранить в памяти и обрабатывать изменяемые данные, обращаясь к ним просто по имени

Константы разных типов хранят значения неизменяемых данных

$$y = \sin ax + b/2$$

- переменные для хранения значений параметров y, a, x, b
- операторы для ввода значений a, x, b и вывода результата y
- специальный оператор для расчета значения синуса
- удобная и понятная для человека запись вычислений

Пример на Паскале

```
program Equation;
var a, b, x, y : real;
begin
   readln(a, b, x);
   y := sin(a*x) + b / 2;
   writeln('y = ', y);
end.
```

Пример на C++

```
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
  double a, b, x, y;
  cin >> a >> b >> x;
  y = sin(a*x) + b / 2;
  cout << "y = " << y << endl;
  return 0;
}
```

Общее определение алгоритма

Алгоритм – набор инструкций (команд), определяющих порядок действий исполнителя для решения некоторой задачи

Свойства алгоритма

Дискретность – алгоритм определяется как последовательность отдельных инструкций (команд)

Определенность – каждая команда должна быть однозначно понятна исполнителю

Конечность – алгоритм должен завершать свою работу за конечное время

Процесс разработки алгоритма

Математическая модель – совокупность переменных и уравнения (соотношения) для них, описывающие некоторое явление

Процесс разработки алгоритма

Информационная модель – описание данных и их свойств

Для построенной информационной модели необходимо разработать алгоритм, написать программу и проверить ее работоспособность

Программа – алгоритм, записанный на каком-либо языке программирования

Трансляция программы

Транслятор (**компилятор**) – это программа, на вход которой подается текст алгоритма на языке программирования – **исходный модуль**, а на выходе (после трансляции) получается программа на машинном языке – **объектный модуль**.

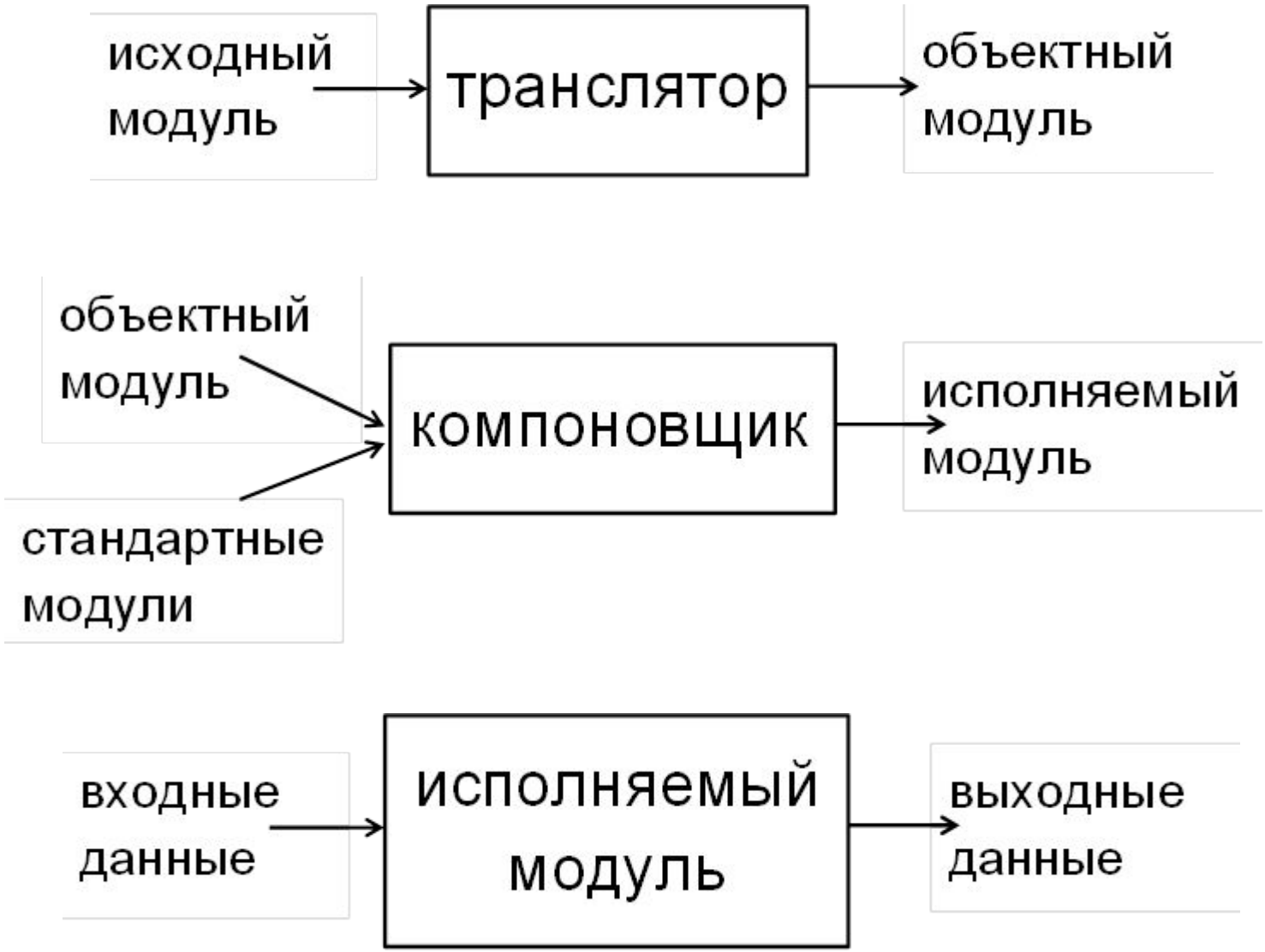
Транслятор действует по строго формальным правилам: если транслируемая программа содержит хотя бы одну формальную (*синтаксическую*) ошибку, то трансляция не может завершиться!

Компоновка программы

Компоновщик (**редактор связей**) – это программа, которая объединяет объектный модуль и необходимые для выполнения дополнительные модули (поддержка ввода/вывода, стандартные функции и т.д.) в единый **исполняемый модуль**

В некоторых системах программирования трансляция и компоновка выполняются как один шаг

Общая схема работы



Синтаксис и семантика языка программирования

Синтаксис – формальные правила, которым должна соответствовать программа на некотором языке программирования

Семантика – смысл отдельных элементов программы, написанной по правилам синтаксиса

Тестирование программы

После успешной трансляции в программе могут остаться смысловые (**семантические**) ошибки.

Для их обнаружения программу необходимо **тестировать**: т.е. подготовить некоторые входные данные, подать их на вход при выполнении программы, и сравнить **получившиеся** выходные данные с **ожидаемыми** выходными данными.

Чтобы выявить все возможные ошибки, тестировать необходимо на большом количестве различных входных данных!

Цель тестирования: выявить как можно больше возможных ошибок!

Тестирование программы

Тестирование методом **черного ящика**: при создании тестов внутренняя структура программы не принимается во внимание.

Тестирование методом **белого ящика**: тесты создаются на основе внутренней структуры программы, так, чтобы все компоненты программы выполнялись в разных тестах.

Пошаговое тестирование*:* выполнение программы в интерактивном режиме с отслеживанием промежуточных значений переменных.

Исчерпывающее тестирование программы

Пример: программа сложения двух целых чисел, диапазон каждого из чисел от –2147483648 до +2147483647, всего их 2^{32}.

Количество всех возможных тестов, т.е. различных пар слагаемых 2^{64} = 18446744073709551616.

При 10^9 проверках в секунду для исчерпывающего тестирования потребуется более 500 лет!

Закон Э. Дейкстры: «*Тестированием можно доказать наличие ошибок в программе, но никогда – их отсутствие*»

Среда разработки

В настоящее время для разработки программ созданы специальные интерактивные системы поддержки (**IDE**).

Они предоставляют пользователю:

- текстовые редакторы для работы с исходным кодом
- трансляторы и средства сборки программ
- библиотеки функций, классов и визуальных компонент
- инструменты для тестирования
- и многое другое