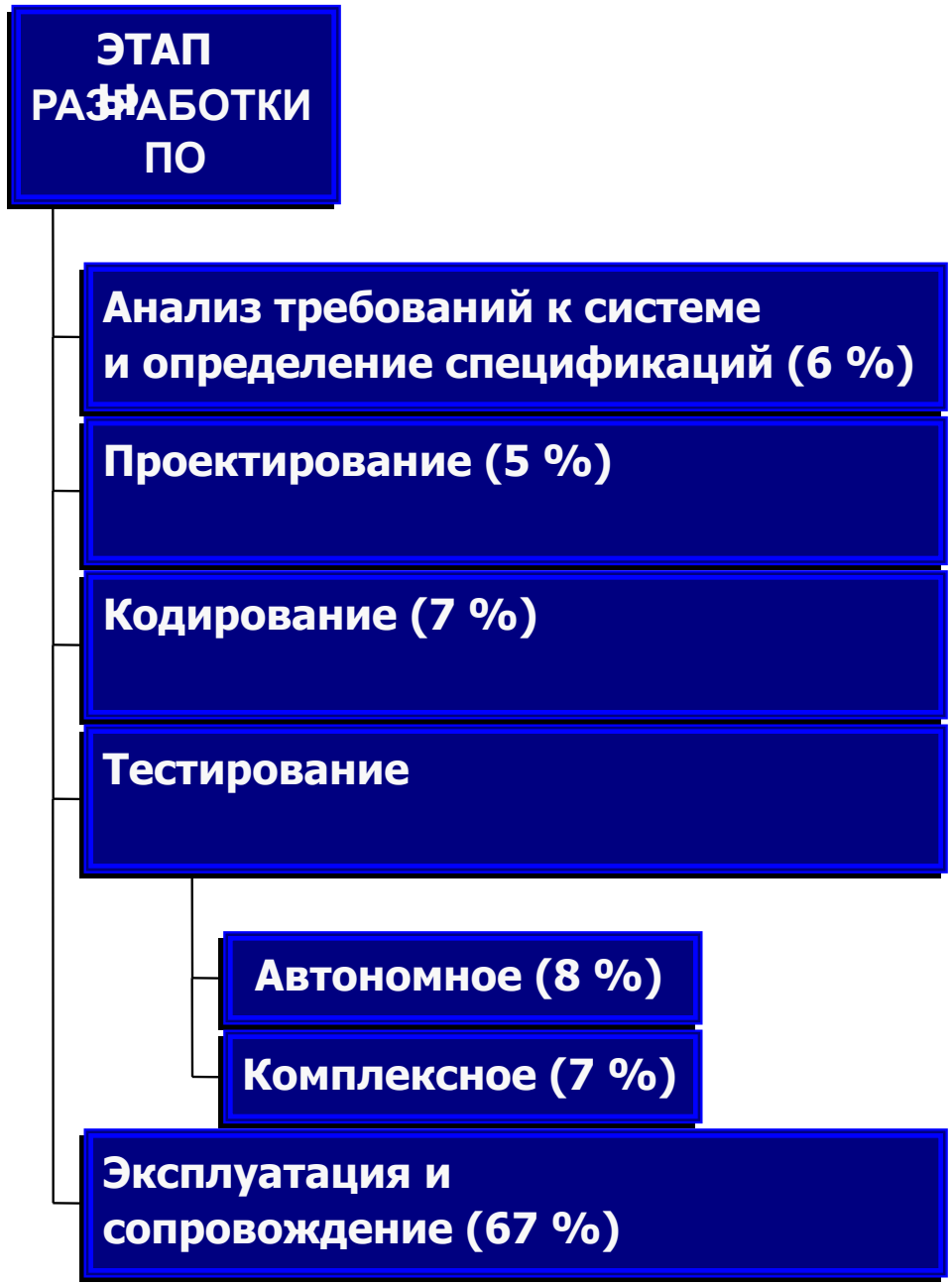
Программное обеспечение (пакет программ) – группа взаимосвязанных и взаимодействующих программ, предназначенных для решения любой задачи из конкретной области (например, для моделирования технологических процессов, для выполнения графических работ). Программы какого-либо пакета рассчитаны на совместное использование в различных комбинациях друг с другом.



На первом этапе первичным документом является **постановка задачи**, объединяющая:

- общую характеристику задачи (назначение, экономическая эффективность, структура объектов);
- описание входных данных (структура и способ поступления);
- описание выходных данных;
- существующие к данному моменту алгоритмы получения выходных данных на основе входных;
- источники разработки (откуда появилась вся информация).

Результатом первого этапа является **техническое задание**, в котором, как правило, содержатся
название системы, цели создания, характеристика области применения, требования к системе в целом (интерфейс, особые требования к отдельным модулям, безопасность, и т.д.), информационная база (структура базы данных, с которой будет взаимодействовать программная система), программное обеспечение (обоснование выбора языка программирования), техническое обеспечение, описание данных для тестирования.

На основании технического задания формируются *спецификации* – описание количества и режимов работы модулей, их взаимодействия.

На этапе **проектирования** разрабатываются *алгоритмы*, задаваемые спецификациями, и формируется *общая структура* будущей программы путем детальной проработки последовательности ее действий.

Для разработки алгоритмов сложных программ используется метод **пошаговой детализации**, при котором процесс преобразования исходных данных в результат вначале представляется в виде последовательности небольшого числа простых этапов (задач). На следующем шаге задачи разбиваются на последовательность подзадач следующего уровня и т.д. Детализация заканчивается, когда каждый отдельный этап может быть относительно простым способом записан на выбранном языке программирования, или представляет собой известную задачу, для которой уже имеется готовая программа.

Формальное описание алгоритмов осуществляется, например, с использованием *языка схем* или *псевдокода*.

Кодирование представляет собой реализацию разработанных алгоритмов, составление по ним текстов программы с использованием конкретного языка программирования. Включает процесс *трансляции* – перевода программы в последовательность машинных команд (машинный код).

При **автономном тестировании** каждый модуль проверяется отдельно. При этом программная среда модуля имитируется с помощью программы управления тестированием, содержащей фиктивные программы вместо реальных подпрограмм, к которым имеется обращение из данного модуля.
При **комплексном тестировании** производится совместная проверка групп программных компонентов.

В процессе тестирования происходит *оптимизация* системы (разгрузка участков повторяемости – циклов, замена сложных операций на более простые, экономия памяти и т.д.).

Большая часть расходов, затрачиваемых в течение жизненного цикла системы, приходится на *эксплуатацию и сопровождение*.

Причины выпуска новых версий (модификаций) ПО:

- необходимость исправления ошибок, выявленных в процессе эксплуатации;
- необходимость совершенствования, например, улучшения интерфейса или расширения состава, выполняемых функций;
- изменение среды (появление новых технических средств и/или программных продуктов).

Языки программирования - это тщательно составленные последовательности слов, букв, чисел и мнемонических сокращений, используемые для общения с компьютером.

Машинная команда состоит из кода операции и адресной части. Код операции указывает вид выполняемого действия (сложить, вычесть и т.д.). Адресная часть содержит адреса (номера) ячеек памяти, в которых расположены операнды, и адрес ячейки, куда следует поместить результат

Языки ассемблера отличаются от машинного кода тем, что коды операций заменены буквенными обозначениями (например, *ADD* – сложить, *MOV* – переслать данные) и вместо номеров ячеек используются символические адреса.

Ассемблер – программа, преобразующая мнемонику языка ассемблера непосредственно в двоичные представления машинных команд.

Языки ассемблера – машинно-зависимые языки, языки низкого уровня, в которых одна команда соответствует одной машинной команде.

Языки программирования, имитирующие естественные языки и способные на основании одного предложения строить несколько команд компьютера, принято считать **языками высокого уровня**.

Варианты языков – подмножества, расширения, диалекты.
Подмножество – это версия языка, включающая только часть возможностей полного языка. **Расширение** – это расширенная версия, дополненная новыми свойствами, делающими язык более разносторонним. **Диалекты** содержат незначительные изменения, которые способствуют либо настройке языка на специальное применение, либо выявлению сильных сторон конкретного компьютера. Диалекты несовместимы и с исходным языком, и с другими диалектами.

Критерии оценки (выбора) языка программирования:

- легкость чтения программ (особенно важна с точки зрения эксплуатации и сопровождения программ);
- легкость создания программ в выбранной области;
- надежность (т.е. насколько программа соответствует своему предназначению в любых условиях).

Основные **характеристики** языка: простота; структурированность; доступные типы и структуры данных; синтаксис языка; поддержка абстракции; выразительность; проверка совместимости типов; обработка исключительных ситуаций.

Главными элементами **императивных языков** программирования являются переменные, которые моделируют ячейки памяти; операторы присваивания, основанные на операции пересылки данных; а также итеративная форма повторений (циклы). В основе использования этих языков лежит технология *структурного программирования*, базирующаяся на *процедурной декомпозиции.*

FORTRAN (FORmula TRANslator). Считается первым компилируемым языком высокого уровня. В основном используется для программ, выполняющих естественно-научные и математические расчеты. Одной из особенностей этого языка (вплоть до диалекта FORTRAN 90) было то, что типы и ячейки памяти для всех переменных фиксировались до выполнения программы. В процессе выполнения программы новые переменные не вводились и распределение памяти не производилось. Отсюда невозможность создания рекурсивных подпрограмм, трудность реализации динамических структур данных.

COBOL (COmmon Business Oriented Language, 1960 г.). Структура и словарь близки к обычному английскому языку. Является основным языком в США для обработки данных в таких учреждениях как банки и страховые компании. Раздел данных – сильная сторона языка COBOL, тогда как раздел процедур – относительно слабая (в частности, отсутствует достаточное число функций).

BASIC (Beginner's All-purpose Symbolic Instruction Code). Исходный BASIC имел только 14 операторов и один тип данных – числа с плавающей точкой. Был очень популярным языком для микрокомпьютеров в конце 70-х и начале 80-х годов, т.к. был легок для изучения начинающими и не требователен к ресурсам компьютера. Важнейшим аспектом была его ориентация на использование удаленного доступа к компьютеру посредством терминала. Ранние версии не были средством написания серьезных программ значительного размера. Возрождение языка BASIC произошло в начале 90-х годов с выходом языка *Visual BASIC* (Microsoft, 1991).

ALGOL (ALGOrithmic Language). Язык ALGOL – результат попытки создания универсального языка. В этом языке была формализована концепция типов данных, реализована идея составных операторов. В диалекте ALGOL 60 была введена концепция блочной структуры, что позволяло программистам локализовать части программы, вводя разные области видимости. Имелась возможность передавать параметры подпрограммам по значению и по имени, а также создания рекурсивных процедур. Были доступны динамические массивы (ALGOL 68), т.е. массивы, диапазон индексов которых задавался переменной, которая могла менять свое значение во время работы программы. Свыше 20 лет ALGOL оставался единственным официальным средством представления алгоритмов в научной литературе. Недостатки: некоторые его свойства были слишком гибкими, в основной версии отсутствовали операторы ввода-вывода.

Императивные языки, являющиеся *потомками* языка ALGOL.

Pascal. Обеспечивает возможность создания больших программ, поддерживая их строгую логическую структуру. Для коротких программ может оказаться слишком громоздким. Считается важнейшим инструментом для обучения методам структурного программирования.

C. Отличная замена ассемблеру для низкоуровневого программирования, с одной стороны, и применения принципов структурного программирования – с другой. Стал очень популярен благодаря многим решениям, сделавшим запись программы на С весьма компактной. В целом С – очень мощный и в то же время изящный язык. Одной из особенностей является отсутствие полной проверки типов, что порождает гибкость языка в сочетании с ненадежностью.

Ada (1980 г.). Происходит от Pascal, но заметно сложнее его. Содержит средства выделения в отдельные модули объектов данных, обеспечивает поддержку использования абстракции данных в структуре программы. Содержит обширные средства обработки исключительных ситуаций. Программные блоки в языке Ada могут быть настраиваемыми. Язык также обеспечивает параллельное выполнение особых программных блоков, которые называются заданиями.

Modula-2 (1981 г.). В сравнении с Pascal добавлена поддержка модулей. Имеются абстрактные типы данных, возможность использования процедур как типов, низкоуровневые средства системного программирования и сопрограммы.

Объектно-ориентированные языки программирования в наибольшей степени способны реализовать технологию *объектно-ориентированного программирования*, которая основана на *объектной декомпозиции*.

Smalltalk (1980 г.). Основные идеи происходят от первого ООЯ SIMULA 67. Программными модулями языка Smalltalk являются объекты – структуры, объединяющие локальные данные и набор операций (методы), которые доступны другим объектам. Метод определяет реакцию объекта на определенное сообщение, соответствующее данному методу. Абстракциями объектов являются классы. Экземпляры классов – объекты программы. Имеется иерархия классов. Подклассы наследуют функциональные возможности и переменные родительского класса, имея возможность добавлять новые функциональные возможности, а также изменять или скрывать унаследованные.

C++. Представляет собой надстройку над языком С, поддерживающую большинство возможностей, открытых языком Smalltalk (т.е. С++ - это объединение императивного и объектно-ориентированного языков). Поддерживается наследование и множественное наследование, когда класс имеет несколько родительских классов. Операторы в языке С++ могут перегружаться, а подпрограммы - настраиваться. Динамическое связывание обеспечивается функциями виртуального класса. Язык С++ практически полностью совместим с языком С. Недостатки: объемность и сложность.

Смешанные языки, созданные путем введения в соответствующий императивный язык *объектно-ориентированный поддержки*: **Oberon** (1987 г.) **Delphi** (1995 г.), **Ada 95** (1995).

Java. Является непосредственным наследником C++. Отличается от него отсутствием некоторых потенциально ненадежных механизмов, а также тем, что устранены любые, не относящиеся к объектно-ориентированному программированию, средства. Нет возможности написания на языке Java независимых подпрограмм, т.е. все подпрограммы являются методами и определяются в классах. Отсутствует множественное наследование, присущее, в частности, языку C++, – причина нередких ошибок. В языке Java можно создавать параллельные процессы, которые называются потоками. Все объекты в этом языке – динамические. Существует механизм неявного удаления объектов из динамической памяти. Повышение надежности, в частности, связано с отказом от части приведений типов, проверки диапазонов изменения индексов. Особенность языка Java состоит также в использовании особого вида трансляции – *динамической кодогенерации*.

LISP (LISt Processing). Разрабатывался в связи с работами в области искусственного интеллекта. В чистом языке LISP (1958 г.) существовало только два типа структур данных: атомы и списки. Списки представляли собой совокупность узлов, каждый из которых представляет собой два указателя. Первый указывает на представление атома, т.е. на его символьное или числовое значение, а второй – на следующий элемент списка.
Все вычисления в этом языке, который разрабатывался как язык *функционального программирования*, производятся путем применения функций к аргументам. Итеративные процессы определяются с помощью рекурсивных функций.

Prolog (PROgramming LOGig, 1972 г.) – язык *логического программирования*, основная идея которого состоит в использовании формальной логической записи для сообщения компьютеру вычислительных процессов. Программирование в нем является непроцедурным. Программы не устанавливают точно, как должен вычисляться результат, а только описывают его форму. Транслирующая система сама должна выбрать нужный порядок выполнения команд, который приведет к желаемому результату. Логическое программирование было использовано, главным образом, в системах управления реляционными базами данных, экспертных системах и обработке текстов на естественных языках.

Поколения языков программирования (Generation Language, GL)

- **1 GL** (время появления – 40-50 гг.). Языки машинных команд. Первый ассемблер, который позволял задавать названия команд в символическом виде и указывать числа не только в двоичном, но и в десятичном или шестнадцатеричном формате.
- **2GL** (конец 50-х – начало 60-х гг.). Символический ассемблер, включающий понятие переменной.
- **3GL** (60-е годы). Универсальные языки высокого уровня (FORTRAN, COBOL, ALGOL, Pascal, C и др.) Современные среды программирования (например, Delphi, Visual BASIC, Java Development Kit) включают поддержку объектно-ориентированной технологии, набор визуальных инструментов для скоростной разработки приложений (RAD), библиотеку визуальных компонент (VCL) и другие инструменты.
- **4 GL** (начало 70-х г.). Непроцедурные языки, ориентированные на конкретную область применения (в отличие от универсальных языков 3GL), в частности, на работу с базами данных. Команды языков 4GL ближе к обычному языку, нежели у 3GL:

  FIND ALL RECORDS WHERE NAME IS "SMITH"

  Типичный пример языка 4GL – SQL (Structured Query Language).
- **5 GL** (наст. время). В том числе, системы создания программ, ориентированные на непрограммиста.

Трансляция - перевод программы, написанной на языке программирования, в последовательность машинных команд.

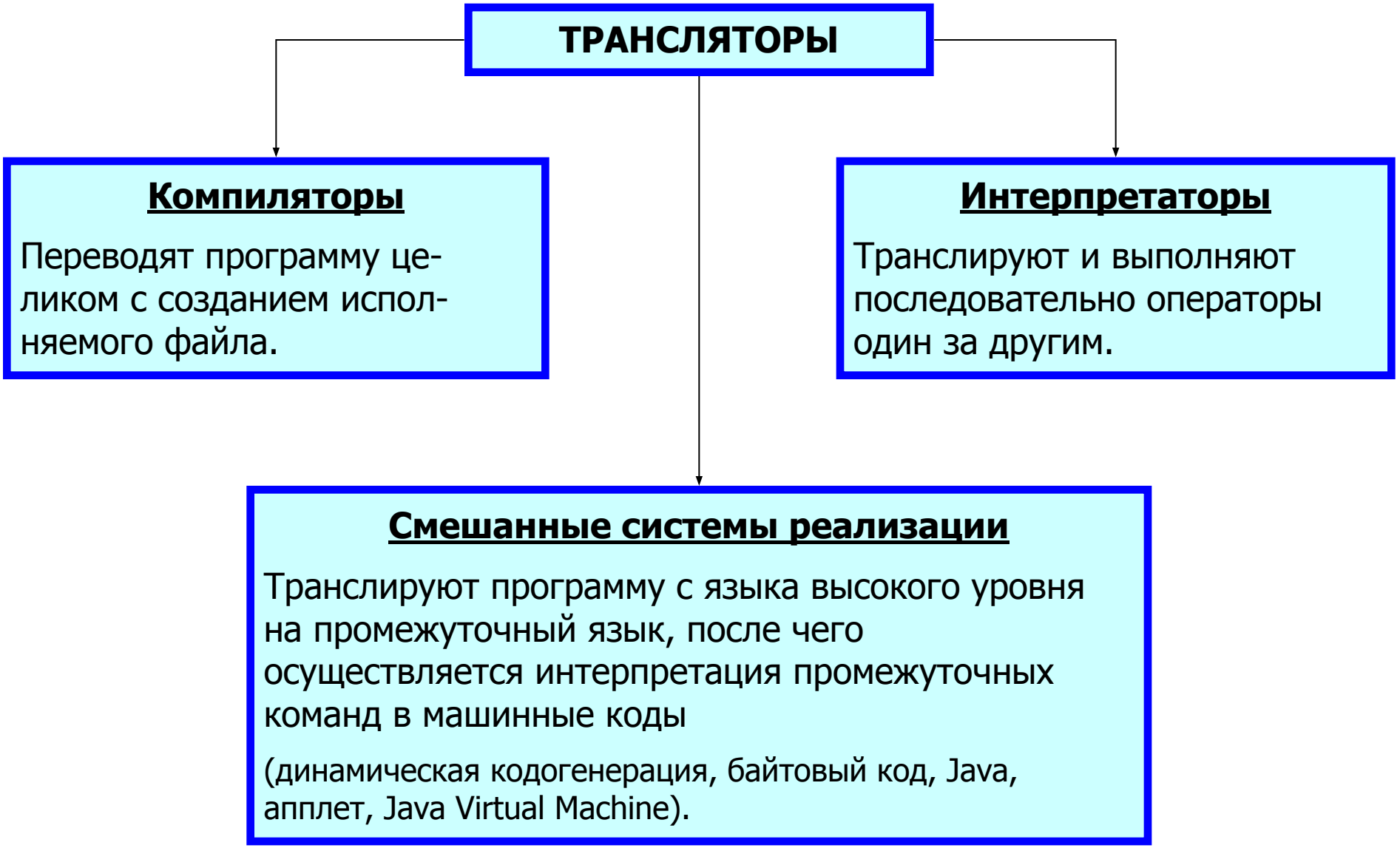
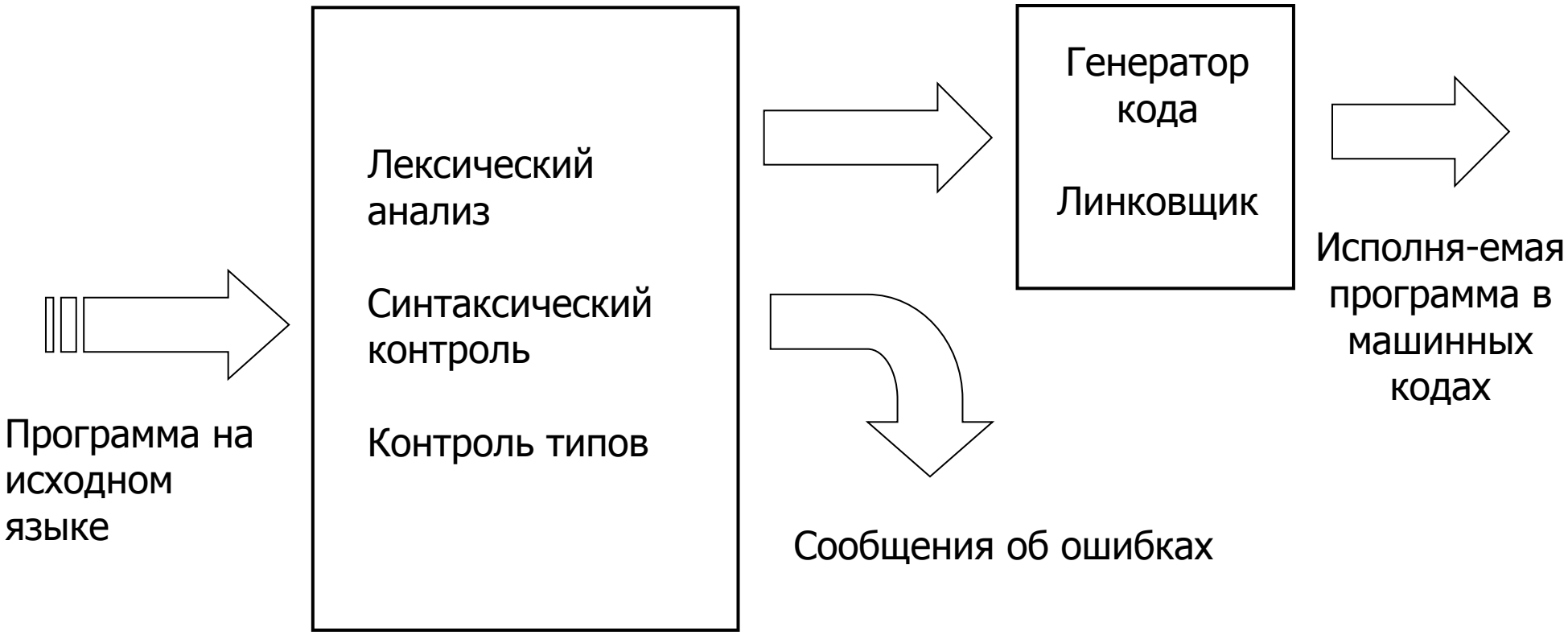



Схема работы компилятора