Программирование

для образовательных программ
Бизнес-Информатика и
Программная Инженерия НИУ
ВШЭ Пермь

Программирование

- Лектор:
  - Викентьева Ольга Леонидовна, доцент каф. ИТБ, к.т.н. (oleovic@rambler.ru, ovikenteva@hse.ru,)
- Практики:
  - Викентьева Ольга Леонидовна, доцент каф. ИТБ, к.т.н.
  - Марквирер Владлена Дмитриевна, преподаватель каф. ИТБ,
  - Гордеева Ольга Игоревна, старший преподаватель каф ИТБ
  - Шиловских Петр Андреевич, преподаватель каф. ИТБ

Программная инженерия

1 модуль: Основы программирования (типы данных, основные операторы, работа с одномерными массивами).

2 модуль: Процедурно-ориентированное программирование (функции, работа с готовыми классами).

3 модуль: Windows-приложения, основы ООП.

4 модуль: Коллекции. Обобщенное программирование. Объектно-событийное программирование. LINQ.

Бизнес-информатика

1 модуль: Основы программирования (типы данных, основные операторы,).

2 модуль: Процедурно-ориентированное программирование (функции, работа с одномерными массивами)).

3 модуль: работа с готовыми классами C#, Windows-приложения,.

4 модуль: основы ООП

5 модуль: Коллекции. Обобщенное программирование.

6 модуль: Объектно-событийное программирование. LINQ. ОО анализ и проектирование.

Литература

1. Подбельский В.В. Язык C#. Базовый курс.
2. Павловская Т. А. C#. Программирование на языке высокого уровня.
3. Шилдт Г. Полный справочник по C#.
4. metanit.com
5. https://docs.microsoft.com/ru-ru/dotnet/csharp/
6. Лекции по программированию: LMS

Жизненный цикл ПО. Платформа MS.NET

Тема 1

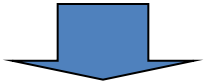
Вопросы

- Жизненный цикл ПО. Модели жизненного цикла
- Парадигмы программирования (анализ)
- Базовые понятия разработки ПО. Способы представления алгоритма (проектирование)
- Языки программирования.
- Тестирование.
- Система программирования.
- Платформа MS.NET
- Структура программы.

Жизненный цикл (ЖЦ) программного обеспечения

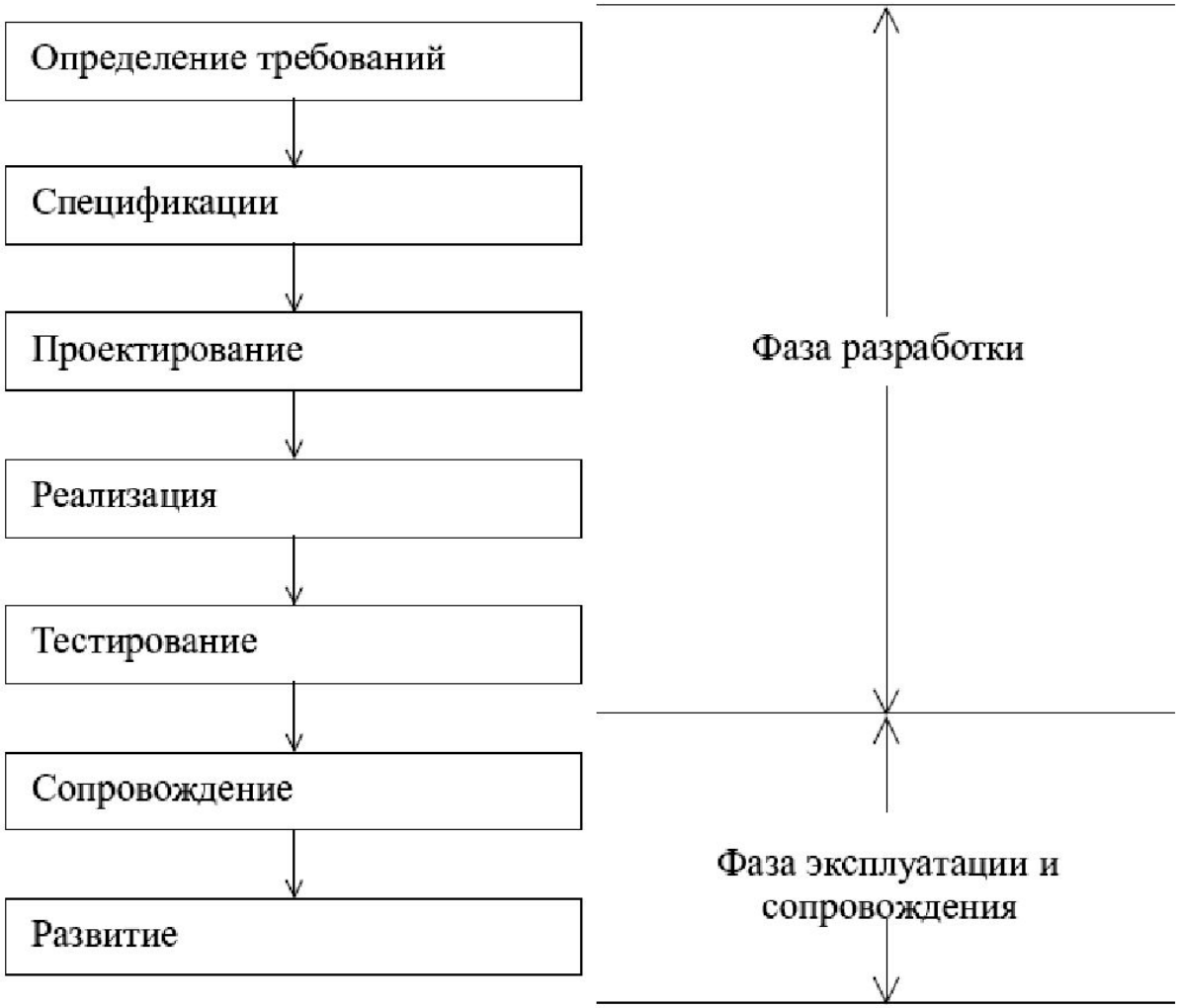
- ЖЦ ПО - совокупность процессов, связанных с последовательным изменением состояния ПО от формирования исходных требований к нему до окончания его эксплуатации.

Международный стандарт ISO/IEC 12207



определяет структуру ЖЦ, содержащую процессы, действия и задачи, которые должны быть выполнены во время создания ПО.

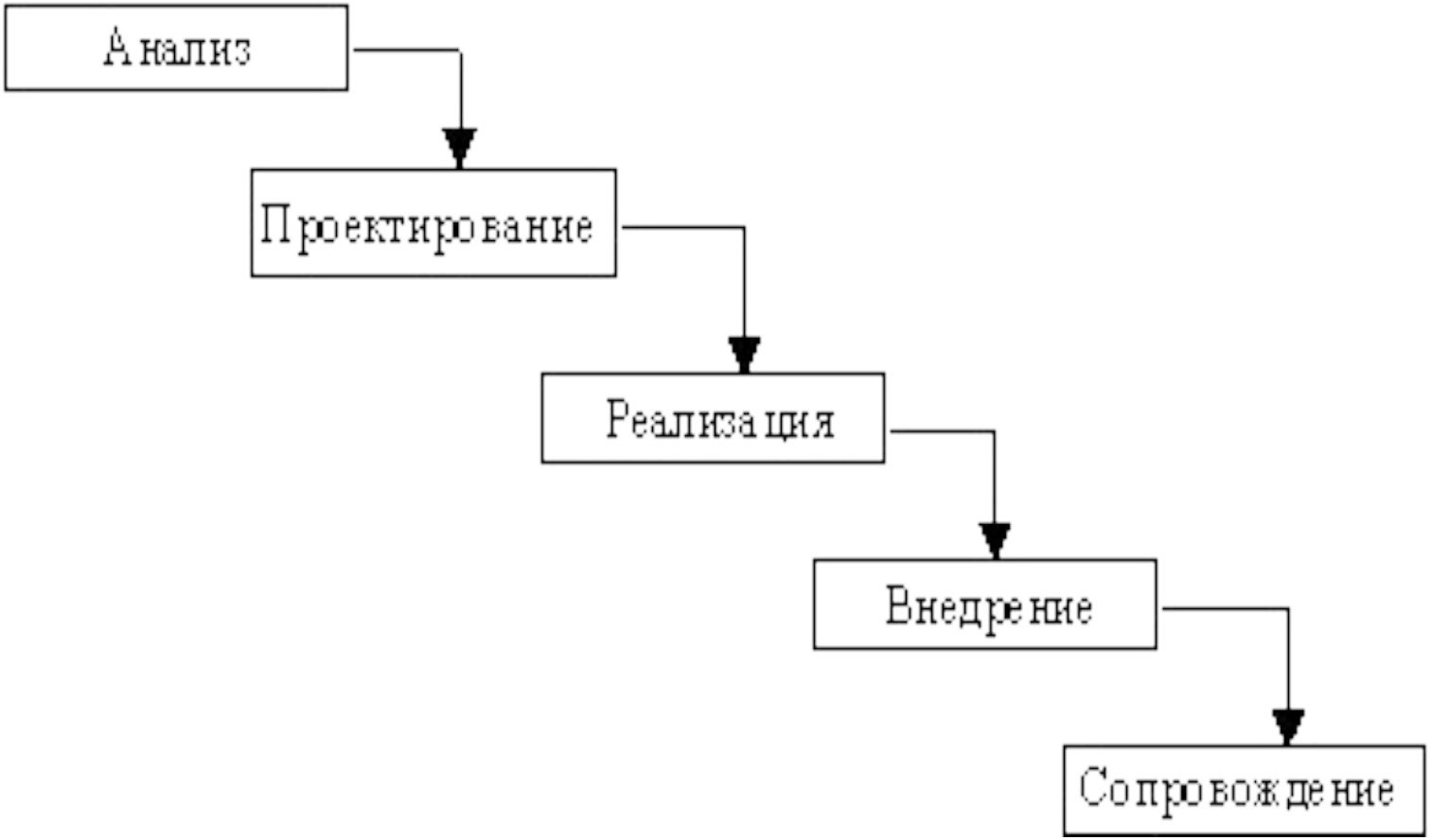
Этапы ЖЦ ПП


Определение требований
Спецификации
Проектирование
Реализация
Тестирование
Сопровождение
Развитие
Фаза разработки
Фаза эксплуатации и сопровождения

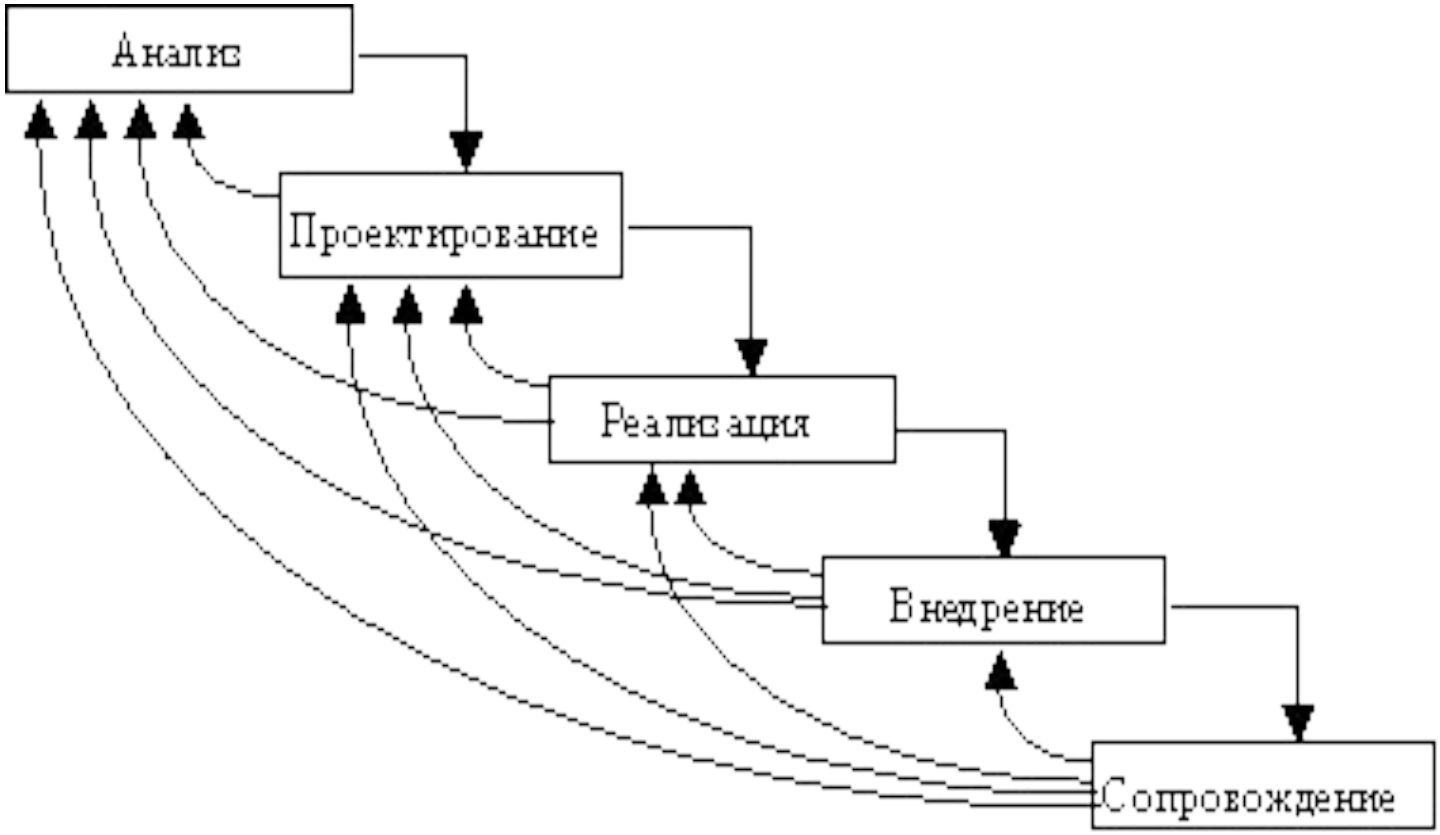
Модели ЖЦ

- Жизненный цикл ПО определяет «что», но не «как» выполнять в процессе программной инженерии.
- Подходы к жизненному циклу ПО могут быть грубо разделены на следующие категории:
  - Каскадная (водопадная) или последовательная.
  - Итеративная и инкрементальная – эволюционная (гибридная, смешанная).

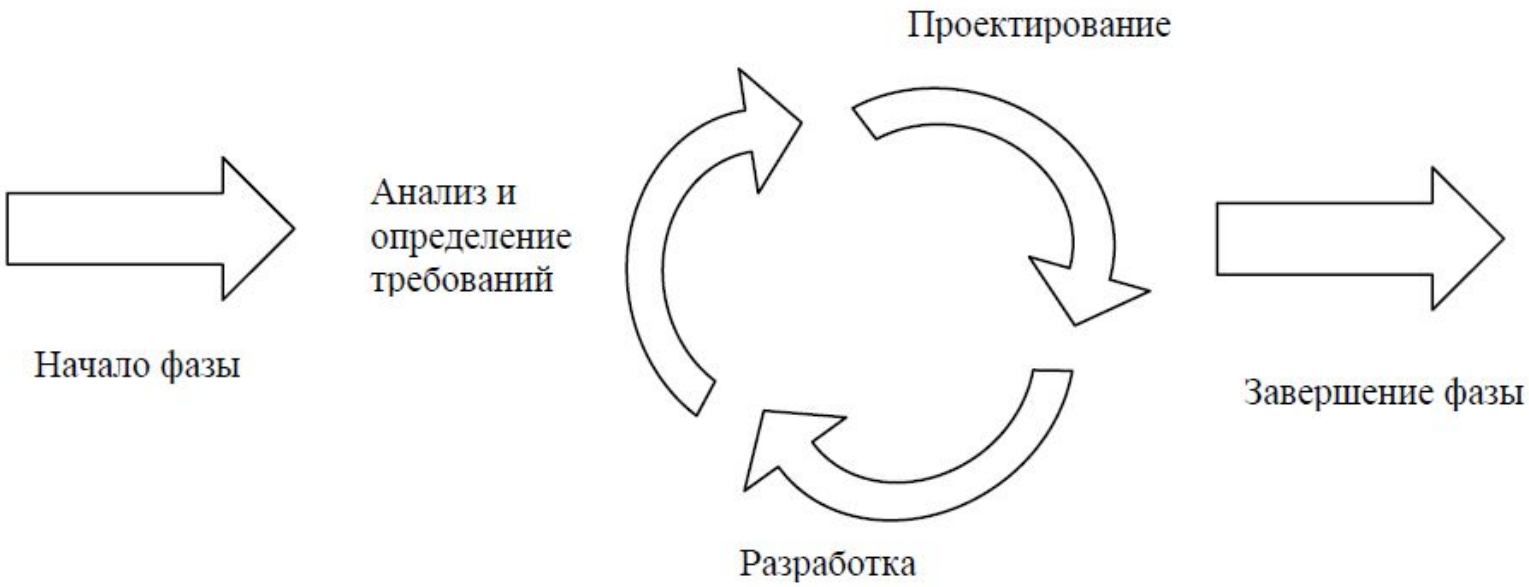
Каскадная схема ЖЦ



Реальный процесс создания ПО



Итерационная схема ЖЦ



Итеративный жизненный цикл предполагает шаги — улучшенные или расширенные версии изделия в конце каждой итерации.

Основные участники процесса создания ПП

- **Заказчик** – определяет требования к разрабатываемой программе (функциональные и нефункциональные).
- **Аналитик** – разрабатывает модель предметной области и спецификацию программы.
- **Проектировщик** – определяет структуру разрабатываемой программы, распределение функциональности между частями программы и их взаимодействие по управлению и обмену данными.
- **Разработчик** – определяет способ реализации требуемой функциональности в каждой из частей программы и разрабатывает код программы на языке, доступном исполнителю.
- **Тестировщик** – ищет ошибки в разработанной программе.
- **Пользователь** – применяет разработанную программу по назначению для получения результатов обработки данных.

Парадигма программирования

- Понятийный аппарат, используемый для разработки моделей предметной области, называют **парадигмой программирования.**
- Существуют:
  - **процедурно-ориентированное программирование,**
  - **объектно-ориентированное программирование,**
  - логическое программирование,
  - функциональное программирование.

Процедурно-ориентированное программирование

- В основе парадигмы лежит понятийный аппарат, отражающий принципы логической организации ЭВМ классической архитектуры.
- В логической модели определяются:
  - входные данные,
  - источники входных данных,
  - выходные данные,
  - потребители выходных данных,
  - данные, подлежащие долговременному хранению (накопители данных),
  - процессы преобразования входных данных в выходные данные.
- Структура программы: набор подпрограмм.
- Взаимодействие подпрограмм организовано по иерархическому принципу. Выполнение программы начинается с главной подпрограммы.

Объектно-ориентированное программирование

- В основе парадигмы лежит представление предметной области в виде множества **объектов**, взаимодействующих между собой.
- **Объект** - мыслимая или реальная сущность, обладающая характерным поведением и отличительными характеристиками и являющаяся важной для данной предметной области.
- Характеристики объекта называют ***атрибутами.*** Атрибуты определяют состояние объекта**.**
- Объект может иметь определенный набор действий (**операций**), которые можно произвести над атрибутами объекта. Набор операций определяет поведение объекта.
- Множество объектов, которые имеют одинаковый набор атрибутов и операций, образуют **класс объектов**.

Базовые понятия разработки ПО

- **Алгоритм** – точное предписание, определяющее вычислительный процесс, идущий от изменяемых начальных данных к конечному результату.
- **Язык программирования** (ЯП) - совокупность средств и правил для представления алгоритма в виде пригодном для выполнения вычислительной машиной.
- ЯП – это искусственные языки, которые имеют очень жесткие правила записи команд. Совокупность этих правил образует **синтаксис** ЯП, смысл конструкций – его **семантику.**
- **Программа** – алгоритм, записанный на ЯП.
- **Программное обеспечение** – набор компьютерных программ, процедур и связанной с ними документации и данных (ISO/IEC 12207).

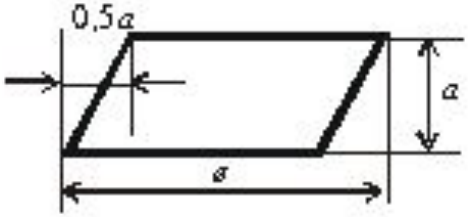
Способы представления алгоритма

- Словесное описание.
- Графическое описание (блок-схема).
- Язык программирования высокого уровня.

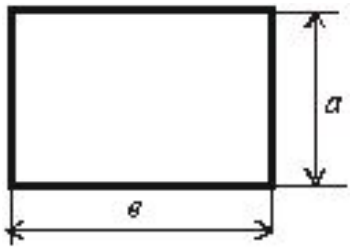
- **Блок-схема** – это последовательность блоков, предписывающих выполнение определенных операций, и связей между этими блоками.
- Конфигурация и размеры блоков, а также порядок графического оформления блок-схем регламентированы **ГОСТ 19002-80 и ГОСТ 19003-80** "Схемы алгоритмов и программ". В январе 1992 года введен новый **ГОСТ 19–701–90**.

Описание символов

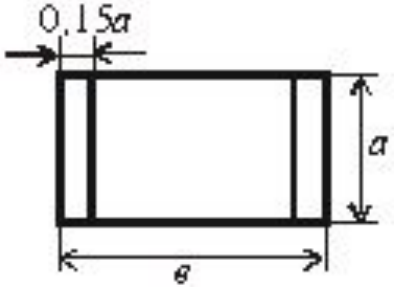
Согласно ГОСТ размеры связаны с двумя величинами: ***а*** и ***в***, где ***а*** – величина, кратная 5, а ***в*** вычисляется по формуле ***в*** = 1,5***а***, допускается ***в*** = 2***а***.



Данные. Символ отображает данные, носитель данных не определен.

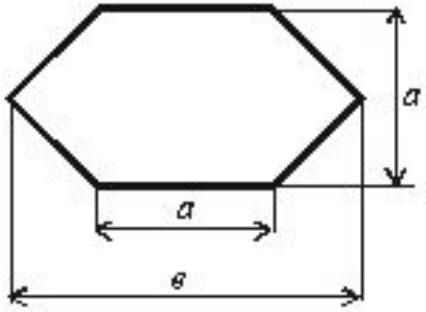


Процесс. Символ отображает функцию обработки данных любого вида (выполнение определенной операции или группы операций, приводящее к изменению значения, формы или размещения информации).

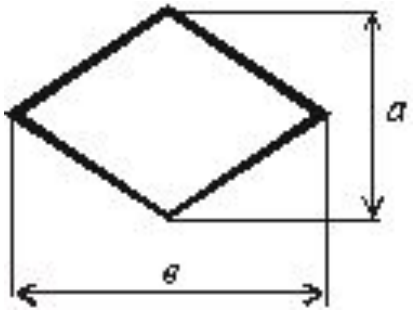


Предопределенный процесс. Символ отображает предопределенный процесс, состоящий из одной или нескольких операций или шагов программы, которые определены в другом месте (в подпрограмме, модуле).

Описание символов



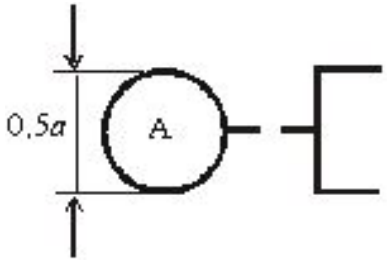
Подготовка. Символ отображает модификацию команды или группы команд с целью воздействия на некоторую последующую функцию.



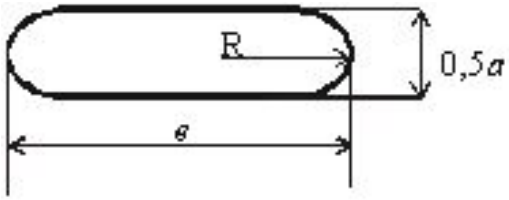
Решение. Символ отображает решение или функцию переключательного типа, имеющую один вход и ряд альтернативных выходов, один и только один из которых может быть активизирован после вычисления условий, определенных внутри этого символа. Соответствующие результаты вычисления могут быть записаны по соседству с линиями, отображающими эти пути.

Описание символов

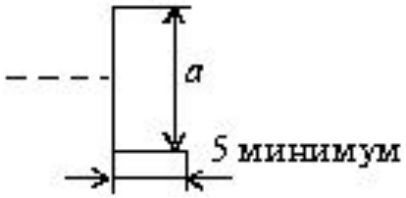
Линия. Символ отображает поток данных или управления. При необходимости или для повышения удобочитаемости могут быть добавлены стрелки-указатели.



Соединитель. Символ отображает выход в часть схемы и вход из другой части этой схемы и используется для обрыва линии и продолжения ее в другом месте. Соответствующие символы-соединители должны содержать одно и то же уникальное обозначение.



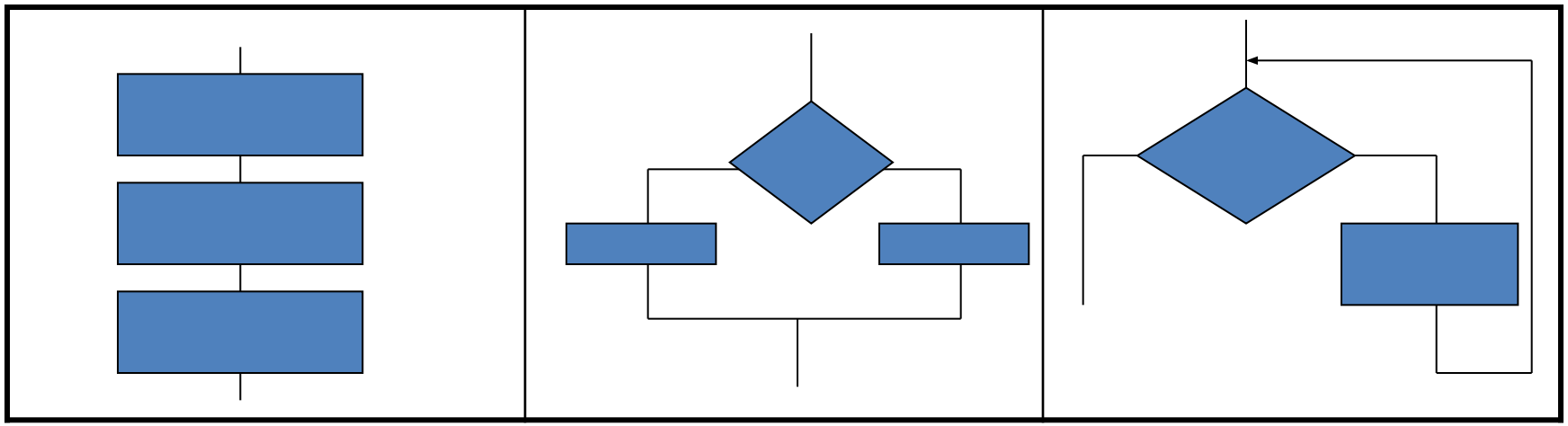
Терминатор. Символ отображает выход во внешнюю среду и вход из внешней среды (начало или конец схемы программы, внешнее использование и источник или пункт назначения данных).



Комментарий. Символ используют для добавления описательных комментариев или пояснительных записей в целях объяснения или примечаний

Основные алгоритмические структуры

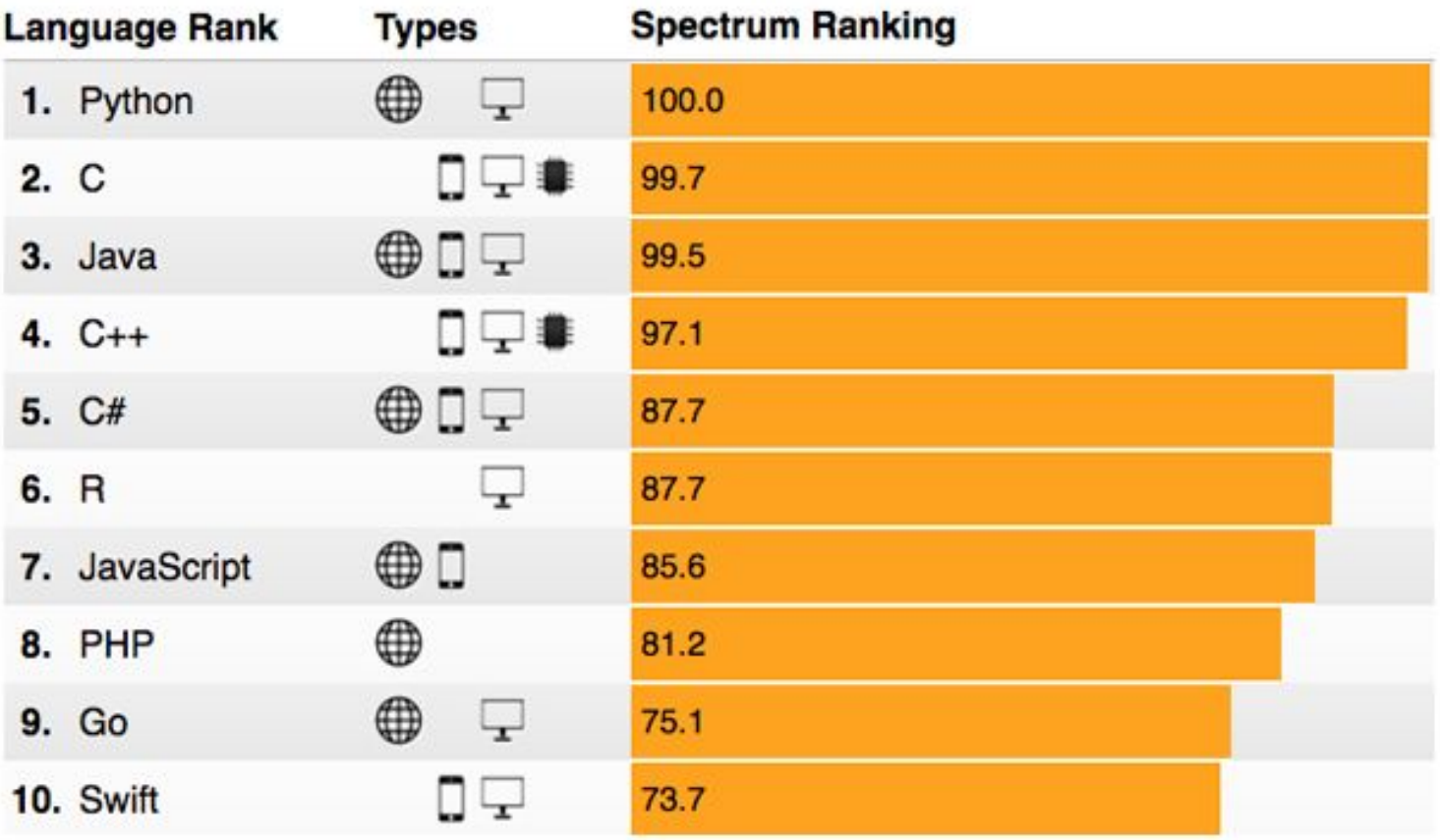
- Любой алгоритм может быть построен из трех базовых структур:
  - следование (последовательность),
  - ветвление,
  - цикл.



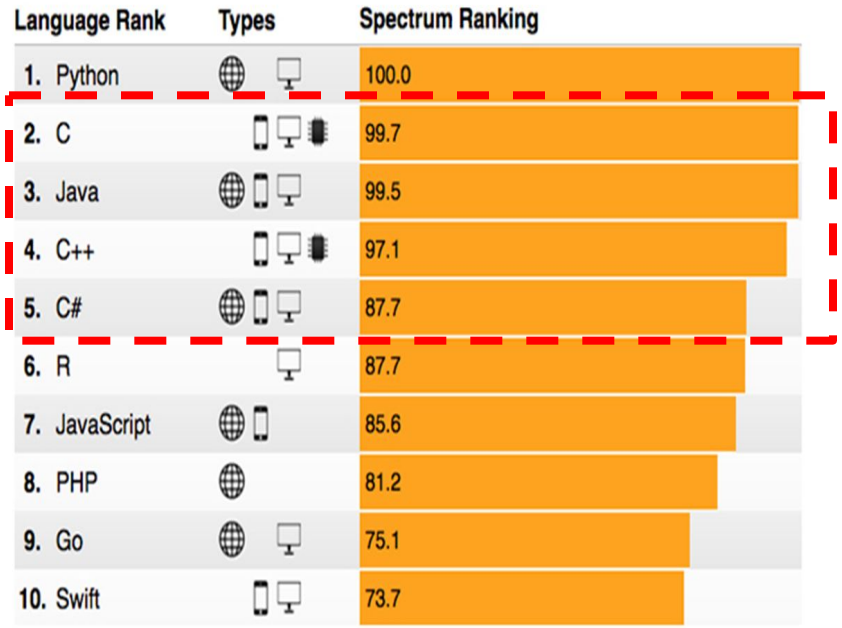
Основные алгоритмические структуры

- **Линейным** называется алгоритм, в котором отдельные предписания выполняются последовательно в порядке записи независимо от значений исходных данных и промежуточных результатов.
- **В разветвляющихся** алгоритмах вычислительный процесс проходит по одной из возможных ветвей.
- **Циклическими** называются алгоритмы, у которых выполнение некоторых операторов осуществляется многократно с одними и теми же или модифицированными (изменяющимися) данными. Циклы бывают:
  - итерационные,
  - арифметические.

Языки программирования



Языки программирования

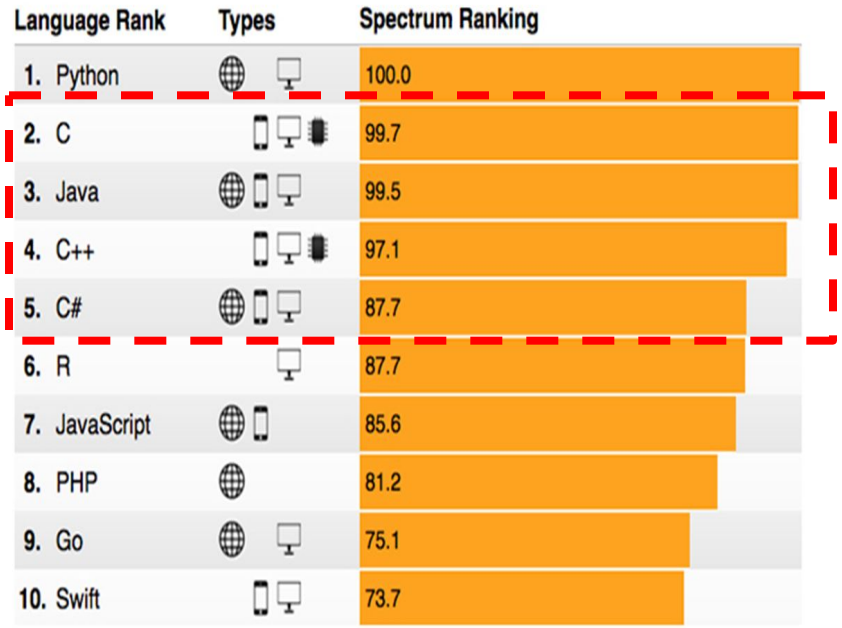


Language Rank	Types	Spectrum Ranking
1. Python		100.0
2. C		99.7
3. Java		99.5
4. C++		97.1
5. C#		87.7
6. R		87.7
7. JavaScript		85.6
8. PHP		81.2
9. Go		75.1
10. Swift		73.7

- C#, Java – используют виртуальную машину.
- Синтаксис похож.
- Управляемая память.

- Отличия:
  - Крос-платформенность.
  - В C# больше новых инструментов (лямбда-выражения,

Языки программирования

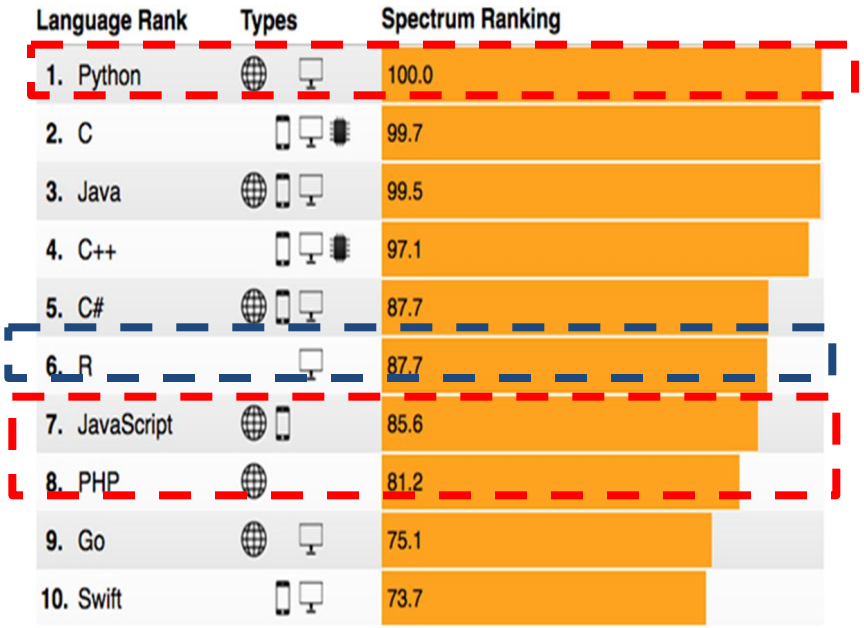


Language Rank	Types	Spectrum Ranking
1. Python		100.0
2. C		99.7
3. Java		99.5
4. C++		97.1
5. C#		87.7
6. R		87.7
7. JavaScript		85.6
8. PHP		81.2
9. Go		75.1
10. Swift		73.7

- C, C++ – компилируемые языки.
- Синтаксис похож.
- Неуправляемая память.

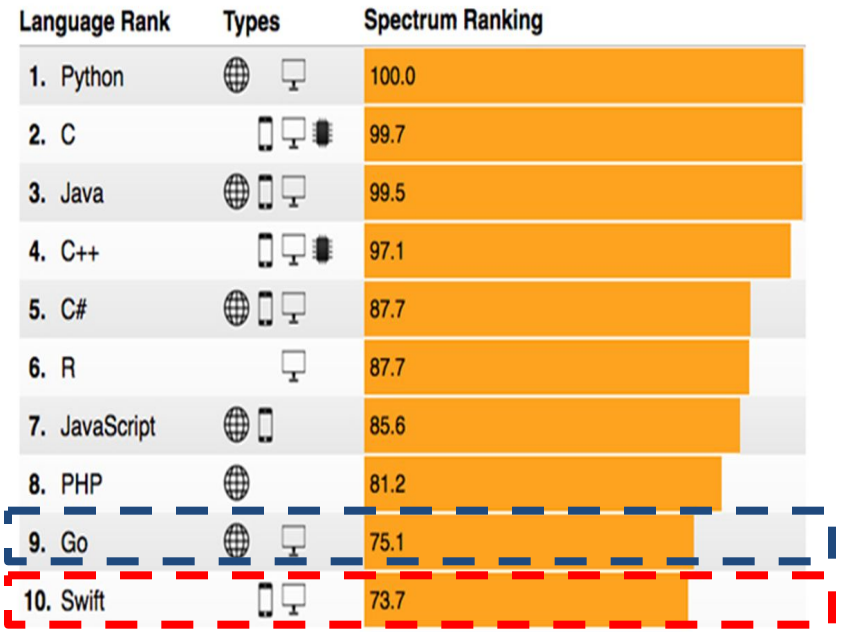
- Отличия:
  - Быстродействие.
  - Realtime (нет ВМ).
  - Небезопасные.

Языки программирования



- Python, JS, PHP:
  - Нет статической типизации.
  - Скрипты (построчная интерпретация).
  - Легко написать маленькую программу.
- R:
  - Применяется для статических расчетов и анализа данных

Языки программирования



Language Rank	Types	Spectrum Ranking
1. Python		100.0
2. C		99.7
3. Java		99.5
4. C++		97.1
5. C#		87.7
6. R		87.7
7. JavaScript		85.6
8. PHP		81.2
9. Go		75.1
10. Swift		73.7

- Go (Golang) — компилируемый многопоточный язык программирования, разработанный внутри компании Google, может рассматриваться как попытка создать замену языкам Си и C++.
- Swift - компилируемый язык программирования общего назначения. Создан компанией Apple в первую очередь для разработчиков iOS и macOS.

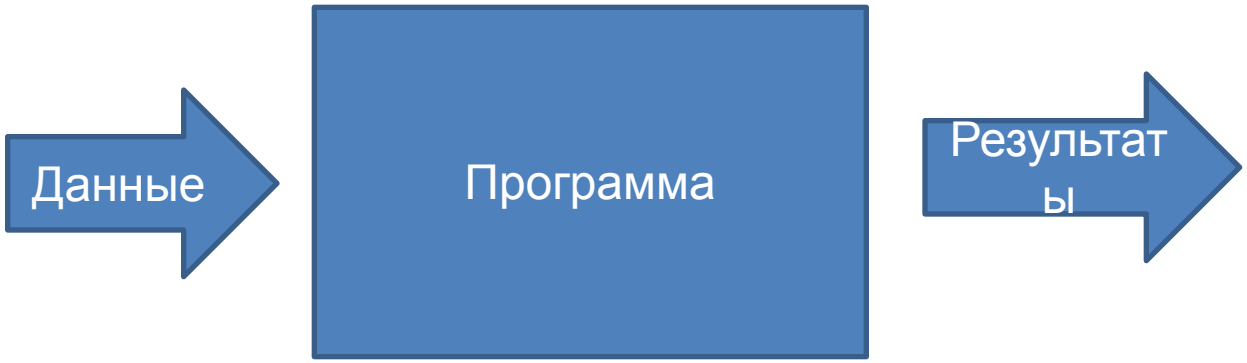
Тестирование

- **Тестирование** – выполнение программы с целью обнаружения в ней ошибок.
- **Отладка** – определение места возникновения ошибки и ее исправление.
- Ошибки в программе бывают трех типов:
  - Синтаксические,
  - Ошибки выполнения программы,

- **Тест** – набор исходных данных, для которых заранее известен результат.
- Если результаты работы теста не совпадают известными значениями, следовательно в программе имеются ошибки.
- **Успешный тест** – тест, который выявил ошибку, т.е. цель тестирования – найти ошибку.
- Отладка заканчивается, когда достаточное количество тестов закончилось **неуспешно**.

Тестирование по стратегии «черного ящика»

- **"Чёрный ящик"** - тестирование функционального поведения программы с точки зрения внешнего мира (текст программы не используется).



Тестирование по стратегии «черного ящика»

- **Тестирование функций**: проверка всех функций, выполняемых программой.
- **Тестирование классов входных данных**: делим данные на классы и считаем, что если программа работает правильно на одном наборе входных данных из этого класса, то она будет работать правильно и на других наборах данных из этого же класса.
- **Тестирование классов выходных данных.**

- **Тестировании области допустимых значений (границ класса):**
  - Нормальные условия (середина класса).
  - Граничные условия.
  - Исключительные условия (выход за границу класса).
- **Тестирование длины набора данных:**
  - Пустой набор.
  - Единичный набор.
  - Слишком короткий набор.
  - Набор минимально короткой длины.
  - Нормальный набор.
  - Набор максимально возможной длины.
  - Слишком длинный набор.

- **Тестирование упорядоченности** (для сортировок и поиска экстремумов):
  - Данные неупорядочены.
  - Данные упорядочены в прямом порядке.
  - Данные упорядочены в обратном порядке.
  - В наборе имеются повторяющиеся значения.
  - Экстремальное значение находится в середине набора.
  - Экстремальное значение находится в начале набора.
  - Экстремальное значение находится в конце набора.
  - В наборе имеются совпадающие экстремальные значения.

Тестирование по стратегии «белого ящика»

- Тестовые данные получаются путем анализа логики работы программы.
  - **Покрытие операторов**: каждый оператор должен быть выполнен хотя бы один раз.
  - **Покрытие ветвей**: каждая ветвь в программе выполняется хотя бы один раз.
  - **Покрытие путей**: каждый путь должен быть пройден хотя бы один раз.

Тестирование по стратегии «белого ящика»

- Покрытие операторов:

```
a=0;
if(x>5) a=10;
b=x/a;
```

- Покрытие ветвей (решений):

```
a=5;
while(a>x) a=a-1;
b=1/a;
```

- Покрытие путей:

Каждый путь должен быть пройден хотя бы один раз..

Проверка сложных условий

- **Критерий покрытия условий**: каждое простое условие получает значение истина.

```
if(a<b)||(c==0) d=1; else d=1/c;
```

- **Критерий покрытия решений/условий** объединяет вместе критерий покрытия ветвей и критерий покрытия условий.

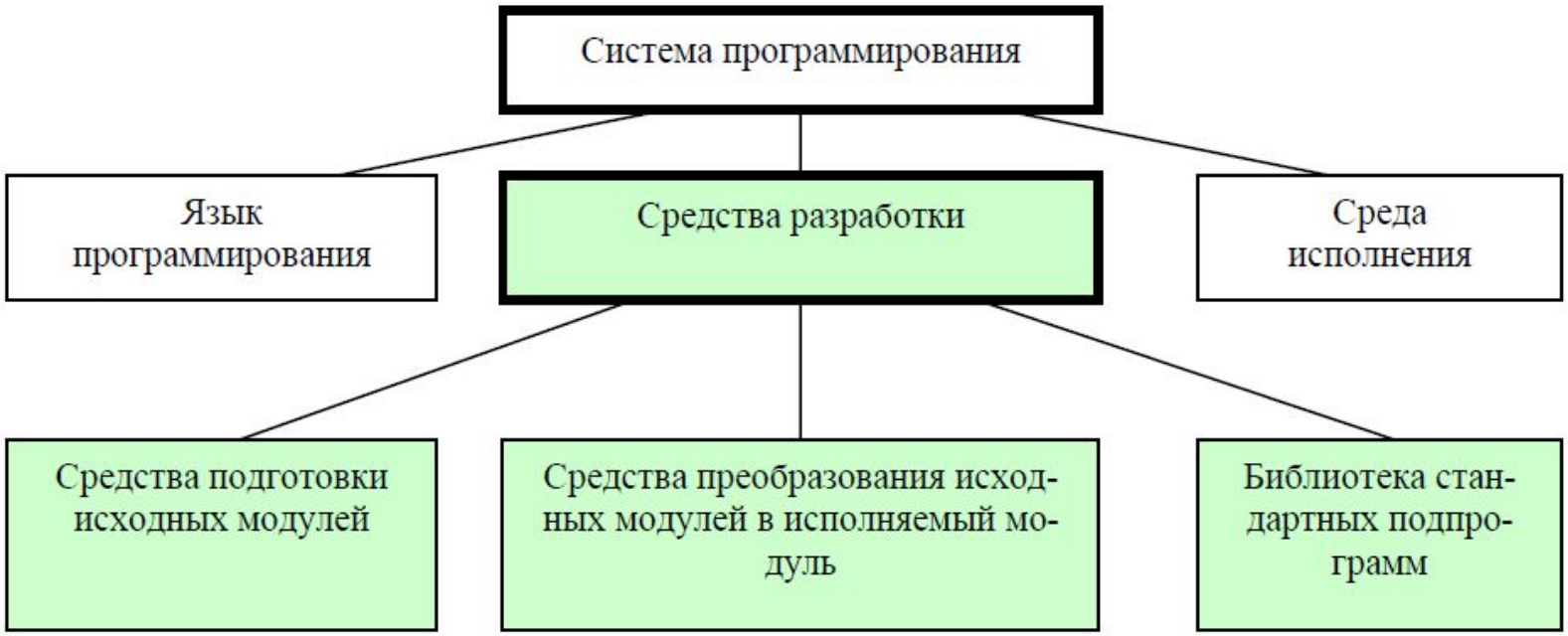
```
if(a==0)||(b==0)||(c==0) d=1; else d=1/(a+b);
```

- **Критерий комбинаторного покрытия условий:** хотя бы один раз должна выполняться любая комбинация простых условий.

Минимальное грубое тестирование

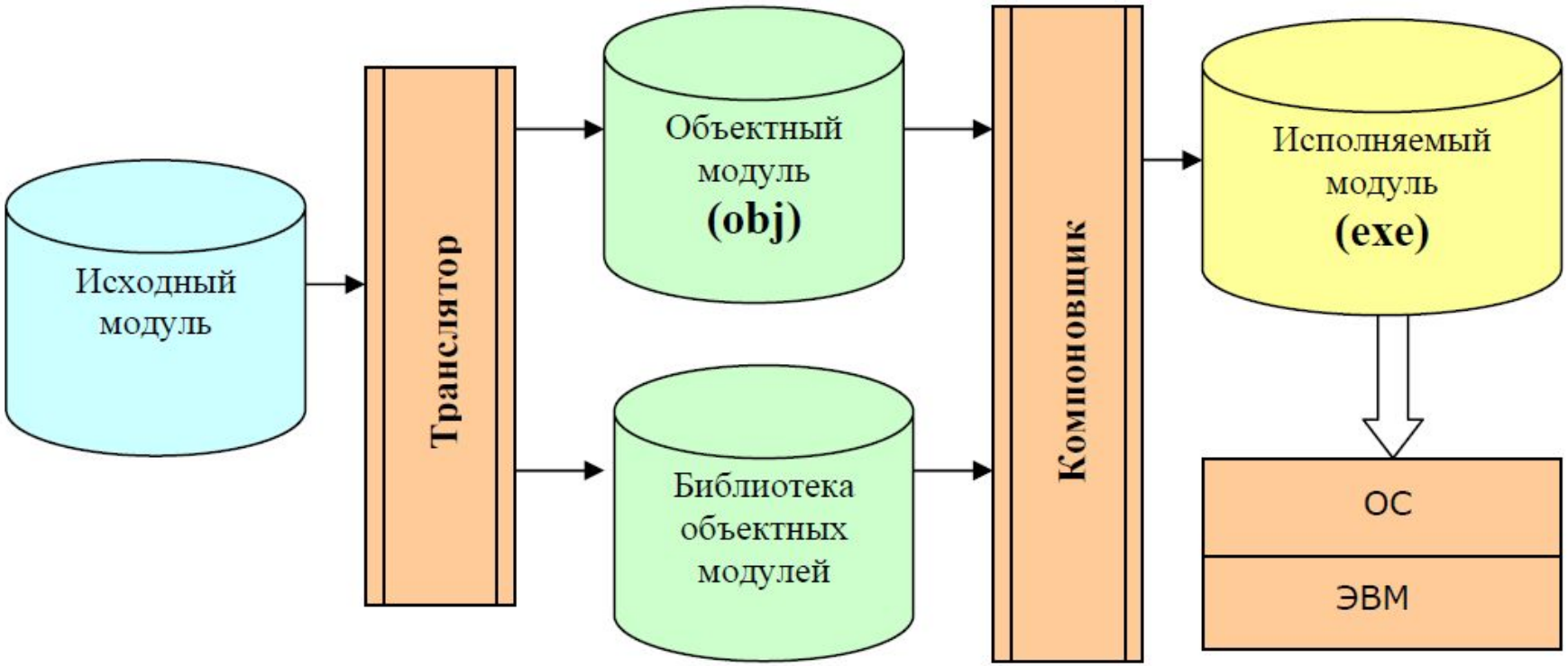
- **МГТ** =критерий покрытия решений/условий + дополнительные требования по проверке циклов:
  - Каждая ветвь в программе выполняется хотя бы один раз.
  - Каждое простое условие получает значение истина.
  - Каждый цикл проверяется при нулькратном, однократном и многократном повторении.

Система программирования



- **Система программирования** - это язык программирования и совокупность программных средств, поддерживающих разработку и исполнение программ, написанных на этом языке.
- Для выполнения программа должна быть загружена в **среду исполнения**.
- Модуль, содержащий программу на языке высокого уровня, называется **исходным модулем.**
- Модуль, содержащий программу в виде, готовом для загрузки в среду исполнения, называется **исполняемым модулем.**
- Различают две основные схемы преобразования исходного модуля в исполняемый модуль: **трансляция и интерпретация**

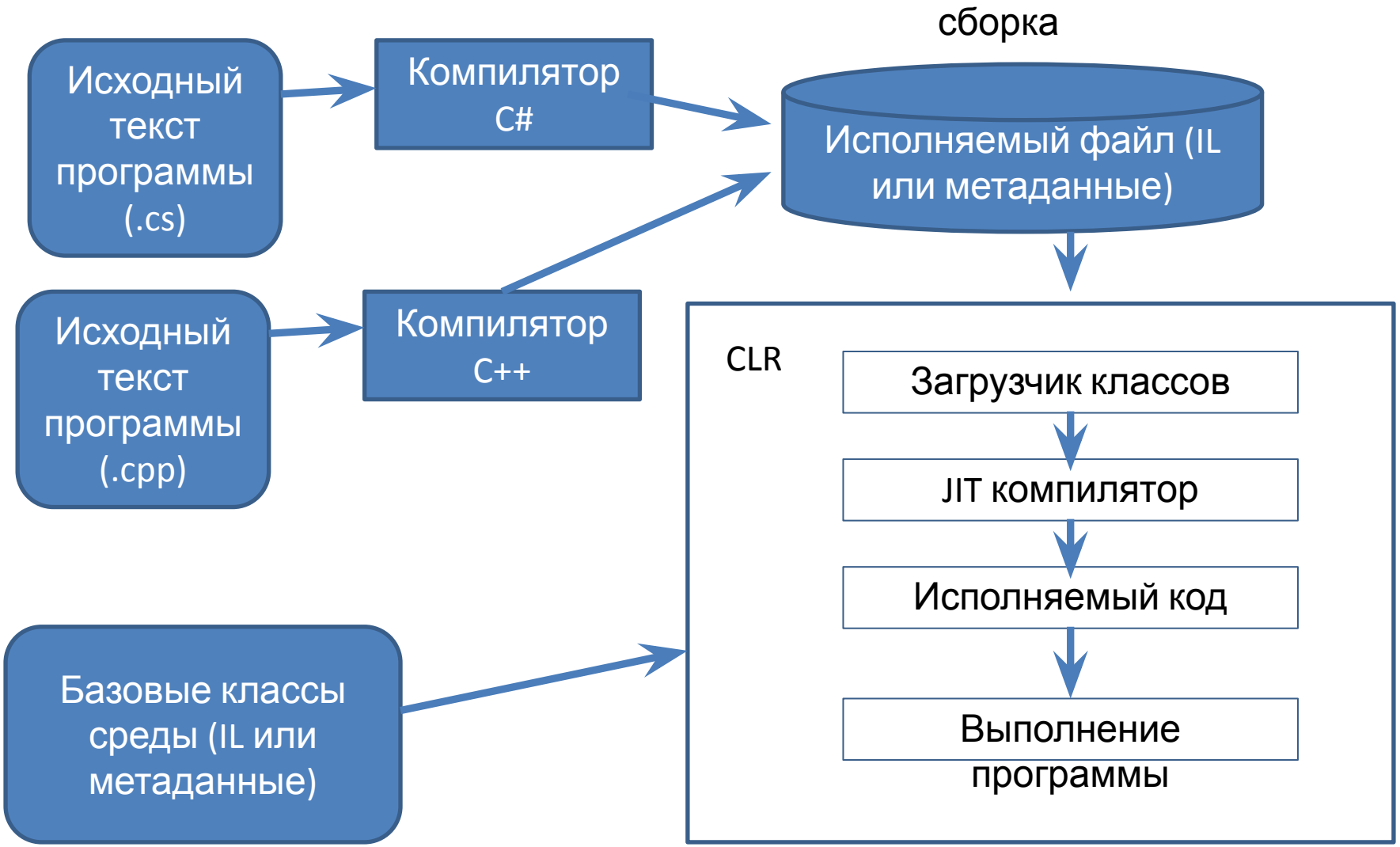
Трансляция


Исходный модуль
Транслятор
Объектный модуль (obj)
Библиотека объектных модулей
Компоновщик
Исполняемый модуль (exe)
ОС
ЭВМ

Общая характеристика платформы MS.Net

- Платформа MS.Net предназначена для разработки и исполнения приложений различных типов:
  - **автономное консольное приложение** с использованием текстового интерфейса пользователя;
  - **автономное Windows-приложение** с использованием графического интерфейса пользователя;
  - **библиотека классов**, которые предназначены для использования в других приложениях;
  - **Web-приложение**, доступ к которому выполняется через браузер и которое по запросу формирует Web-страницу и отправляет ее клиенту по сети;
  - **Web-сервис** – компонент, методы которого могут

Выполнение программы в .NET

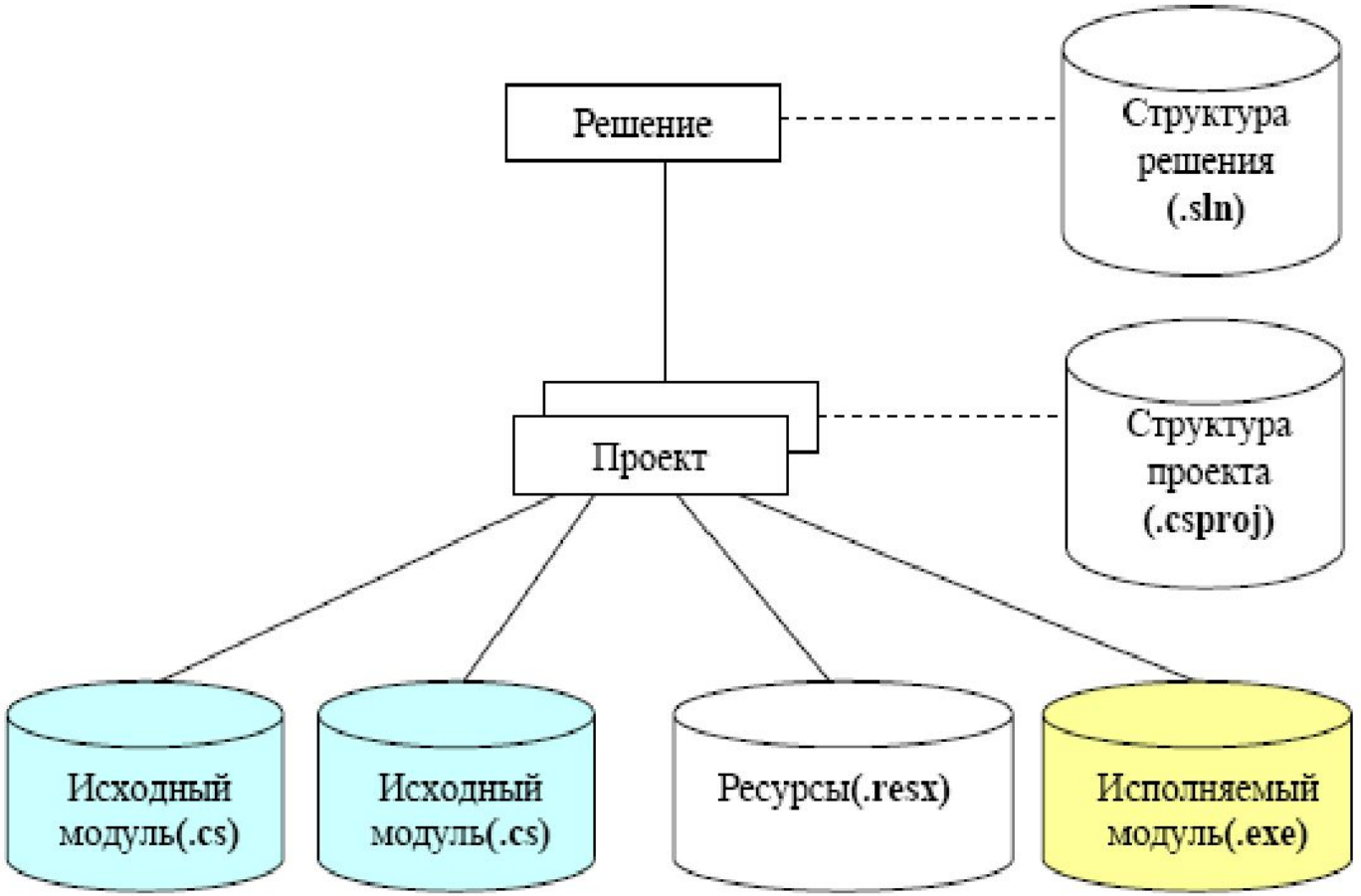


Основные понятия

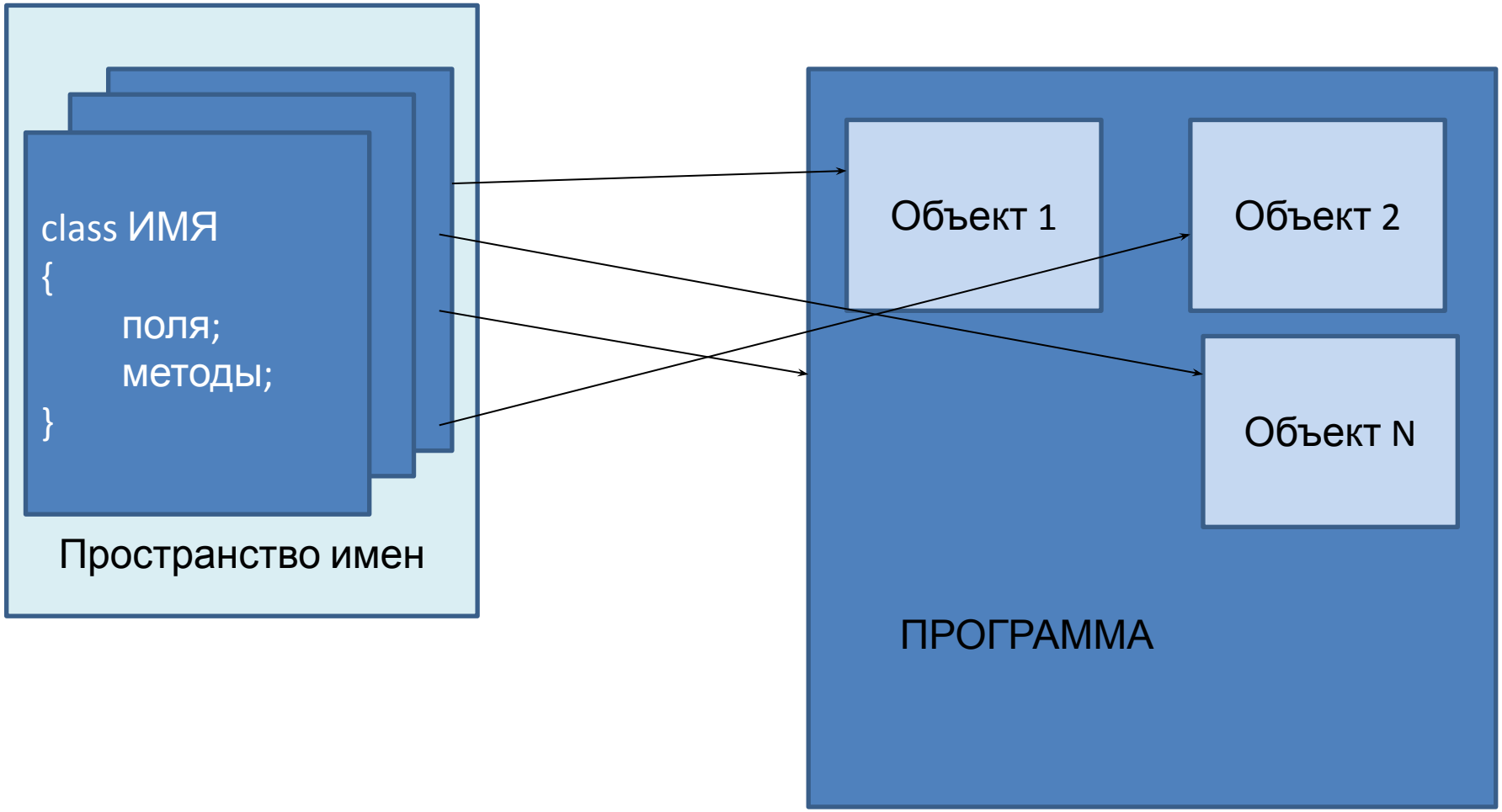
- MSIL или IL (Microsoft Intermediate Language) – промежуточный язык, который не содержит команд, зависящих от языка, ОС и типа компьютера.
- CLR (Common Language Runtime) – общеязыковая среда выполнения, выполняет программу на языке IL. Может быть реализована для любой ОС.
- JIT (just in time) – компилируются только те части программы, которые нужно выполнить в данный момент.
- Сборка – файл с расширением exe или dll, который содержит код на языке IL и метаданные.
- Метаданные – сведения об объектах, используемых в программе и самой сборке.

- **Управляемый код** - исходный код должен быть переведен на специально разработанный для платформы промежуточный язык MSIL (MS Common Intermediate Language, CIL). Для исполнения кода на промежуточном языке приложения используется специальная программная компонента платформы – общеязыковая среда исполнения CLR(Common Language Runtime).
- **Небезопасный код** - исходный код должен быть переведен на язык машинных команд. Машинный код исполняется непосредственно под управлением операционной системы.

Консольное приложение



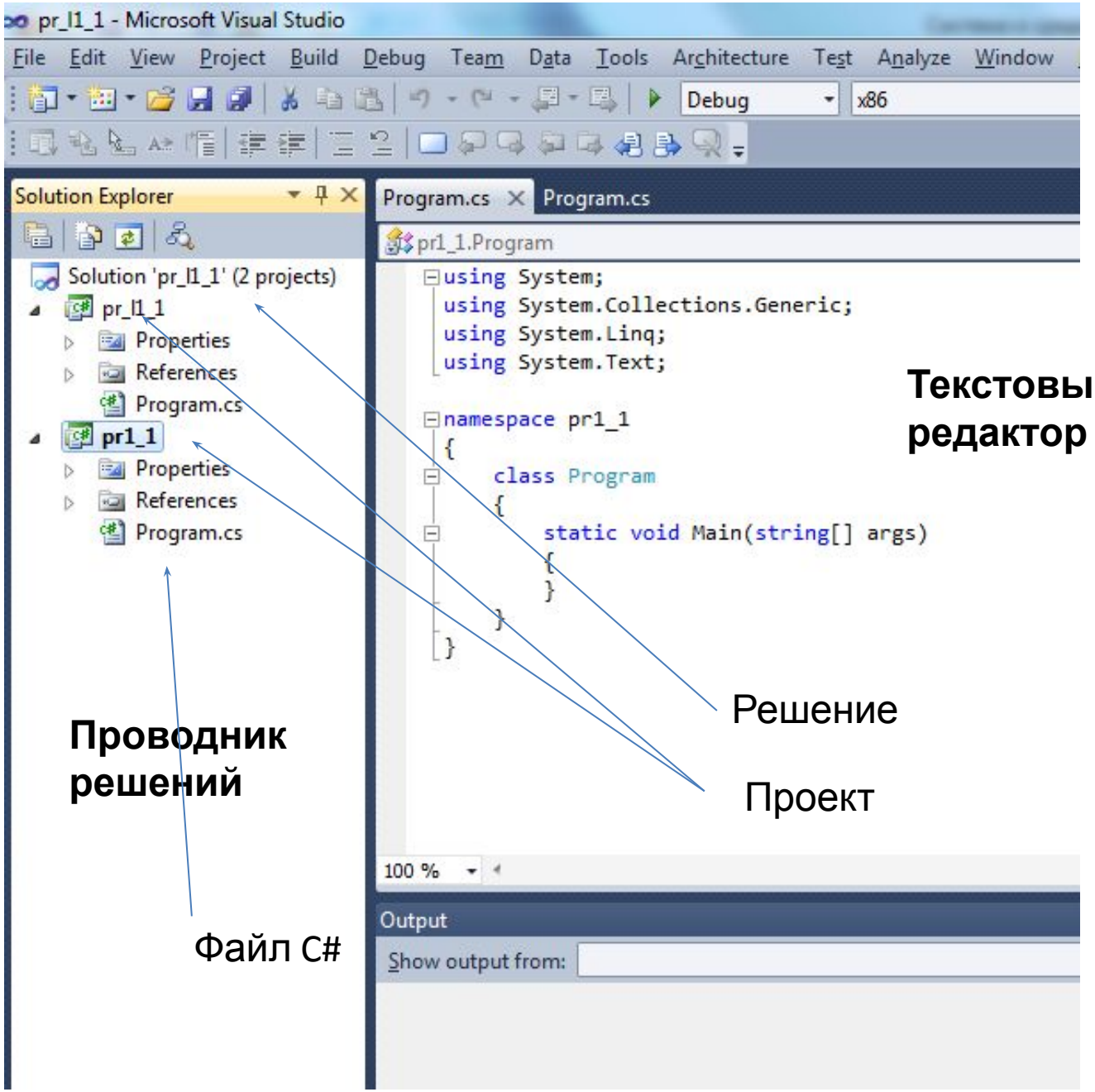
Структура программы на C#



Основные понятия

- Объект – совокупность данных, определяющих состояние объекта и функций, обеспечивающих изменение этих данных.
- Класс - абстрактный тип данных, определяемый пользователем, он задает механизм формирования всех однотипных объектов.
- Метод (функция), определяет действия для изменения данных
- Пространство имен – механизм, обеспечивающий независимость используемых в программе имен.

```
//подключаем пространство имен System
using System;
//пространство имен
namespace App
{
    //класс
    class Program
    {
    //метод
        static void Main()
        {
// текст программы
        }
    }
}
```


pr_l1_1 - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Tools Architecture Test Analyze Window
Debug
x86
Solution Explorer
Solution 'pr_l1_1' (2 projects)
pr_l1_1
Properties
References
Program.cs
pr1_1
Properties
References
Program.cs
Program.cs
Program.cs
pr1_1.Program
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace pr1_1
{
class Program
{
static void Main(string[] args)
{
}
}
}
100 %
Output
Show output from:
Текстовый редактор
Решение
Проект
Проводник решений
Файл C#

Вопросы

- Какой язык программирования изучал?
- Уровень знания языка:
  - Могу написать простую программу с циклами
  - Могу написать простую программу с массивами
  - Могу написать программу с использованием функций
  - Могу написать ОО программу с использованием готовых классов
  - Могу написать ОО программу с разработкой своих классов
  - Могу написать игру с использованием специальной платформы (Unity, Unreal Engine и др)
  - Могу написать и развернуть на сервере веб-приложение
- Использую систему управления версиями при разработке программ. Если да, то какую?
- Участвовал в олимпиадах по программированию (да/нет). Если да, то уровень олимпиады (школа, город, край, РФ и т.д.)
- Планирую участвовать в олимпиадах по программированию и посещать факультатив по олимпиадному программированию (да/нет).