Языки программирования

- 1. Процедурные языки (Императивные)
- 2. Функциональные языки
- 3. Логические языки
- 4. Объектно-ориентированные языки

Кобол

- **Кобол** (*COBOL, COmmon Business Oriented Language*) — один из старейших языков программирования программирования (первая версия в 1959), предназначенный, в первую очередь, для разработки бизнес-приложений.

Кобол

Руководителем проекта по созданию Кобола была Грейс Хоппер (*бабушка Кобола*). Практически с самого своего рождения Кобол является ANSI-стандартизованным языком программирования.

Кобол

Кобол обычно критикуется за многословность и громоздкость, поскольку одной из целей создателей языка было максимально приблизить конструкции к английскому языкуКобол обычно критикуется за многословность и громоздкость, поскольку одной из целей создателей языка было максимально приблизить конструкции к английскому языку. (До сих пор Кобол считается языком программирования, на котором было написано больше всего строк кодаКобол обычно критикуется за многословность и громоздкость, поскольку одной из целей

Кобол

- В то же время, Кобол имел прекрасные для своего времени средства для работы со структурами данных и файлами, он стал первым языком, в котором появился тип данных «запись».[4] Это обеспечило ему долгую жизнь в бизнес-приложениях, по крайней мере, в США.

Кобол

Dr. Dobb’s Journal приводит следующие факты. К 1997 году активно использовалось около 240 миллиардов строк кода на Коболе. Около 90 % финансовых транзакций в мире обрабатывается кодом на Коболе, и 75 % коммерческой обработки данных написано на Коболе. Общая стоимость используемого в настоящее время коболовского кода оценивается в 2 триллиона долларов США. До сих пор ежегодно пишутся миллионы новых строк кода на Коболе.

Алгол

Алгол был разработан в 1958 году, на недельной конференции в ETH (Цюрих, Швейцария) как универсальный язык программирования для широкого круга применений, а затем доработан комитетом, созданным Международной федерацией по обработке информации (IFIP). В комитет вошёл ряд ведущих европейских и американских учёных и инженеров-разработчиков языков.

Алгол

Среди них были: Джон Бэкус — один из создателей Фортрана, Джозеф Уэгстен — впоследствии возглавлял комитет по разработке языка Кобол, Джон Маккарти — автор языка Лисп разработанного одновременно с Алголом, Петер Наур — впоследствии доработал «нормальную форму Бэкуса», завершив разработку БНФ, Эдсгер Дейкстра — нидерландский учёный, впоследствии получивший широкую известность как один из создателей структурного программирования и сторонник математического подхода к программированию.

Алгол

Сначала работа столкнулась с большими трудностями непринципиального характера. Так, например, один из членов комитета вспоминал «десятичную бурю» — крайне резкую дискуссию между американскими и европейскими участниками по поводу того, какой именно символ использовать в качестве разделителя целой и дробной части числа. Американцы стояли за точку, европейцы требовали применять традиционную в Европе запятую, и из-за такой мелочи работа оказалась под реальной угрозой срыва.

Алгол

Чтобы избежать конфликтов по мелким вопросам, было решено, что описание Алгола будет трёхуровневым, включающим уровень описаний, публикаций и реализации. Мелкие вопросы, типа выбора между точкой и запятой или используемого алфавита, были вынесены на второй-третий уровень, что позволило относительно быстро решить принципиальные вопросы. На уровне публикаций, согласованном позже, допускалось использование национальных ключевых слов и стандартов представления данных (в том числе и десятичной точки), уровень реализации определял язык совершенно строго — согласно ему должны были строиться трансляторы.

Алгол

У нового языка нашлись как приверженцы, так и критики. В США Алгол приняли холодно, он был популярен только в академической среде, и то не повсеместно. Те, кто попытался реализовать Алгол, столкнулись с целым рядом сложностей. Так, например, обнаружилось, что ни один из существовавших тогда компьютеров не поддерживал ввод-вывод всех 116 литер, из которых состоял алфавит Алгола.

Алгол

SHARE — американская ассоциация пользователей компьютеров IBM, — потребовала от фирмы реализовать Алгол для своих машин, но появившийся в конце концов компилятор Алгола для IBM OS/360 был крайне неудобен в использовании — вполне естественно, что IBM, вложившая в Фортран огромные суммы, не имела стимула для создания нового продукта, который лишь конкурировал бы со старым. В то же время, недостатки Фортрана вынудили IBM искать ему замену и привели к разработке PL/I — языка-наследника Фортрана, в котором влияние Алгола было весьма заметным.

Алгол

А вот в Европе Алгол приняли с энтузиазмом. Он быстро завоевал популярность в академической среде, повсеместно шла разработка компиляторов, многие из которых, несмотря на сложности реализации, оказались весьма успешными. Алгол распространился от Великобритании до Дальнего востока СССР, став как универсальным языком описания алгоритмов в научных публикациях, так и средством реального программирования.

Алгол

Даже когда язык Алгол почти перестал использоваться для программирования, он ещё долго оставался официальным языком для публикации алгоритмов.

Алгол

- Особенности языка Алгол стали типичными для большинства императивных языков, созданных позднее него. Именно в Алголе появилось представление о программе не как о свободной последовательности команд, а как о блочной структуре, состоящей из чётко описанных и отделённых друг от друга частей. Основной блок программы на Алголе — это сама главная программа. Она содержит свою исполняемую часть, заключённую в блок, ограниченный парой ключевых слов **begin** и **end**, а также описания подпрограмм.

Алгол

- Каждая подпрограмма — это программа в миниатюре, имеющая собственные, описанные внутри неё данные, однозначно определённый интерфейс в виде имени и списка формальных параметров, и блок кода. При этом в блоке могут выделяться подблоки.

Алгол

- Были выделены структурные управляющие конструкции: ветвления, циклы, последовательные участки, исполняющие условно или многократно вложенные наборы операторов, также ограниченные теми же ключевыми словами **begin** и **end**, что дало возможность описывать логику программы без использования безусловных переходов — печально известного оператора goto, провоцирующего на создание запутанных и плохо структурированных программ.

Алгол

Современным программистам подобная структура программы кажется очевидной, кое в чём устаревшей и не всегда удобной (часто критикуются бесконечные **begin** — **end** в программах на Паскале, который унаследовал эту особенность именно от Алгола), но на момент появления Алгола всё это было заметным шагом вперёд. Программы становились регулярными, это давало возможность наращивать их по объёму, сохраняя обозримыми, понятными, доступными анализу и исправлению.

Алгол

Именно на базе Алгола и его языков-потомков были выполнены успешные работы по аналитическому доказательству правильности программ.

Алгол

В Алголе было предложено два способа передачи параметров в подпрограмму — по имени и по значению. Если второй способ возражений не вызывает (он широко используется в абсолютном большинстве языков по сей день), то первый приводил к трудностям реализации компиляторов и появлению труднообнаруживаемых ошибок.

Лисп

Лисп (*LISP*, от англ. *LISt Processing language* — «язык обработки списков»; современное написание: *Lisp*) — семейство языков программирования) — семейство языков программирования, программы) — семейство языков программирования, программы и данные в которых представляются системами линейных списков **символов**. Создатель Лиспа Джон Маккарти. Создатель Лиспа Джон Маккарти занимался исследованиями в области искусственного интеллекта (в дальнейшем ИИ) и созданный им язык по сию пору является одним из

Лисп

Является вторым в истории (после ФортранаЯвляется вторым в истории (после Фортрана) используемым по сей день высокоуровневым языком программированияЯвляется вторым в истории (после Фортрана) используемым по сей день высокоуровневым языком программирования, а также первым из сохранившихся в использовании языков, использующих автоматическое управление памятью и сборку мусора

Лисп

Атомы — это символы и числа. Числа не являются лисповскими символами, поскольку могут иметь только собственное числовое значение и никакого другого. В то же время числа наравне с символами могут входить в списки. Этим и обусловлено объединение этих двух понятий в одну общую категорию.

Лисп

Основная структура данных Лиспа — динамический списокОсновная структура данных Лиспа — динамический список атомов, определяемый рекурсивно как головной объект и присоединённый к нему список-хвост. Поскольку голова списка тоже может быть списком, список, фактически, является формой представления произвольного дерева (сам список верхнего уровня — корень, входящие в него подсписки второго и следующих уровней — узлы, атомы — листья).

Лисп

Для атомов и списков язык использует крайне примитивный скобочный синтаксис: символ представляется своим именем, число — записью его значения, а список — в виде заключённой в круглые скобки последовательности списков и атомов, в которой идущие подряд атомы при необходимости разделены пробелами.

C

Язык программирования Си был разработан в лабораториях Bell Labs в период с 1969 по 1973 годы.
Согласно Ритчи, самый активный период творчества пришёлся на 1972 год. Язык назвали «Си» (C — третья буква латинского алфавита), потому что многие его особенности берут начало от старого языка «Би» (B — вторая буква латинского алфавита).

C

- К 1973 году язык Си стал достаточно силён, и большая часть ядра UNIX, первоначально написанная на ассемблере PDP-11/20, была переписана на Си. Это было одно из самых первых ядер операционных систем, написанное на языке, отличном от ассемблера.

Другие языки

- PL/1
- SNOBOL
- ALGOL 68
- BASIC
- SIMULA 67
- ADA
- PROLOG
- FORTH

Другие языки

- SMALLTALK
- JAVA
- HTML
- PERL
- OCCAM
- C++
- ЛЯПАС

ЛЯПАС

Логический язык проектирования алгоритмов синтеза

Аркадий Дмитриевич Закревский

Структурное программирование

- методология разработки программного обеспеченияметодология разработки программного обеспечения, в основе которой лежит представление программы в виде иерархической структуры блоковметодология разработки программного обеспечения, в основе которой лежит представление программы в виде иерархической структуры блоков. Предложена в 70-х годах XX века Э. Дейкстройметодология

Структурное программирование

1. Любая программа представляет собой структуру, построенную из трёх типов базовых конструкций:

- **последовательное исполнение** — однократное выполнение операций в том порядке, в котором они записаны в тексте программы;
- **ветвление** — однократное выполнение одной из двух или более операций, в зависимости от выполнения некоторого заданного условия;
- **цикл** — многократное исполнение одной и той же операции до тех пор, пока выполняется некоторое заданное условие (условие продолжения цикла).

•В программе базовые конструкции могут быть вложены друг в друга произвольным образом, но никаких других средств управления последовательностью выполнения операций не предусматривается.

Структурное программирование

2. Повторяющиеся фрагменты программы (либо не повторяющиеся, но представляющие собой логически целостные вычислительные блоки) могут оформляться в виде т. н. подпрограмм2. Повторяющиеся фрагменты программы (либо не повторяющиеся, но представляющие собой логически целостные вычислительные блоки) могут оформляться в виде т. н. подпрограмм (процедур2. Повторяющиеся фрагменты программы (либо не повторяющиеся, но представляющие собой логически целостные вычислительные блоки) могут оформляться в виде т. н. подпрограмм (процедур или функций). В этом случае в тексте основной программы, вместо

Структурное программирование

3. Разработка программы ведётся пошагово, методом «сверху вниз».

Структурное программирование

- ***Теорема Бома-Якопини***
- Любую схему алгоритма можно представить в виде композиции вложенных блоков begin и end, условных операторов if, then, else, циклов с предусловием (while) и может быть дополнительных логических переменных (флагов).
  Эта теорема была сформулирована итальянскими математиками К. Бомом и Дж. Якопини в 1966 году и говорит нам о том, как можно избежать использования оператора перехода goto.

Объектно-ориентированные языки

- В основе концепции объектно-ориентированного программирования лежит понятие *объекта* — некой сущности, которая объединяет в себе поля (данные) и методы (выполняемые объектом действия).
- Например, объект человек может иметь поля имя, фамилия и методы есть и спать. Соответственно, в программе можем использовать операторы Человек.Имя:=" Иван" и Человек.Есть(пища).

Объектно-ориентированные языки

- В современных ОО языках используются механизмы:
- *Наследование*. Создание нового класса объектов путём добавления новых элементов (методов. Создание нового класса объектов путём добавления новых элементов (методов). Некоторые ОО языки позволяют выполнять множественное наследование, то есть объединять в одном классе возможности нескольких других классов.

Объектно-ориентированные языки

- *Инкапсуляция*. Сокрытие деталей реализации, которое позволяет вносить изменения в части программы безболезненно для других её частей, что существенно упрощает сопровождение и модификацию ПО.

Объектно-ориентированные языки

- *Полиморфизм*. При полиморфизме некоторые части (методы) родительского класса заменяются новыми, реализующими специфические для данного потомка действия. Таким образом, интерфейс классов остаётся прежним, а реализация методов с одинаковым названием и набором параметров различается.