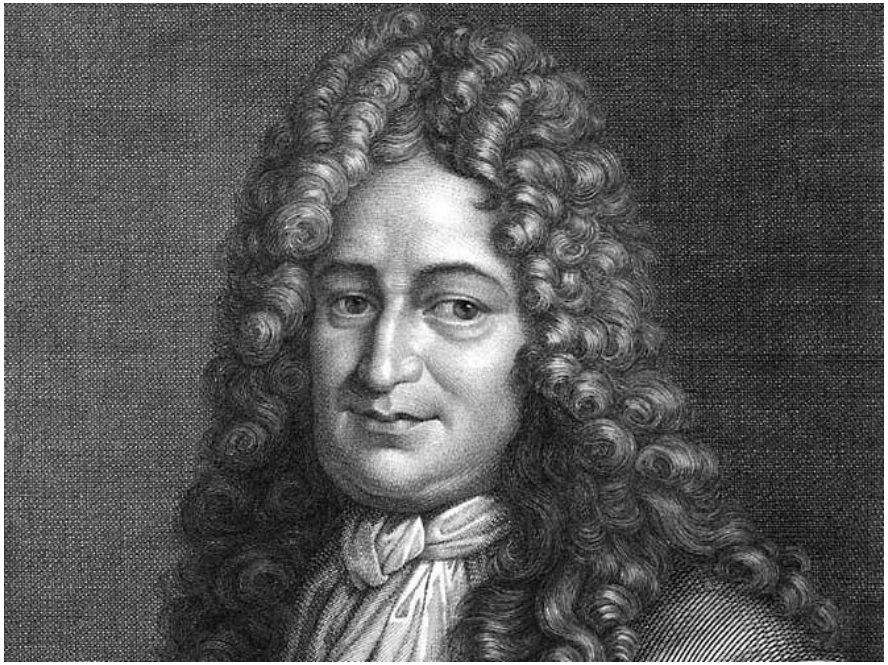
ПОКОЛЕНИЯ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

Теоретические истоки



Готфрид Вильгельм Лейбниц

(1646 – 1716)

немецкий философ, математик, ученый

- изобретатель одной из первых *механических счетных устройств* (1673 г.) – так называемого, колеса Лейбница или шагового барабана. Этот механизм на протяжении нескольких столетий служил основой для конструирования механических счетных устройств типа арифмометра. В 1694 году Лейбниц построил счетную машину Stepped Reckoner, позволявшую складывать, вычитать, умножать, делить, возводить в степень и извлекать корни.
- внес огромный вклад в математическую логику и фактически, заложил ее основы.
- основатель математического анализа и дифференциально-интегрального исчисления, Лейбниц сыграл решающую роль в развитии математики переменных величин. В частности, обобщая свои методы работы с бесконечно малыми величинами, Лейбниц ввел в научный оборот современное понимание таких терминов как: *алгоритм*, *модель*, *функция*, *дифференциал*, *координаты*.

Теоретические истоки

- В течение всей своей жизни Лейбниц работал над идеей *Универсального исчисления* (или Универсальной характеристики) под которым он понимал такую формальную систему, которая бы позволила единым образом описывать понятия и суждения любой тематики и уровня абстракции, отношения и связи между ними.

  Универсальное исчисление Лейбница это попытка ***определения и формализации правил порождения сколь угодно сложных высказываний и конструкций из небольшого набора простых базовых элементов (понятий, аксиом)***.

- Лейбниц разработал и подробно описал аппарат *двоичного исчисления*. Именно этот универсальный аппарат провидчески мыслился Лейбницем как основа Универсального исчисления и как логико-математический язык будущего.

- Описал принципы *машинного моделирования функций человеческого мозга*.

- Вклад в другие науки: физика (ввел понятие кинетической энергии - «живой силы» и сформулировал закон сохранения энергии; идея о превращении одних видов энергии в другие; принцип наименьшего действия; математика (создал комбинаторику как науку); философия (монадология); психология (учение о бессознательном))

Фактически, Лейбниц за 250 лет до возникновения первых компьютеров, определил и блестяще описал основные контуры всей современной информатики и вычислительной техники.

Первые практические решения

Конрад Цузе

(1910 – 1995)

немецкий инженер

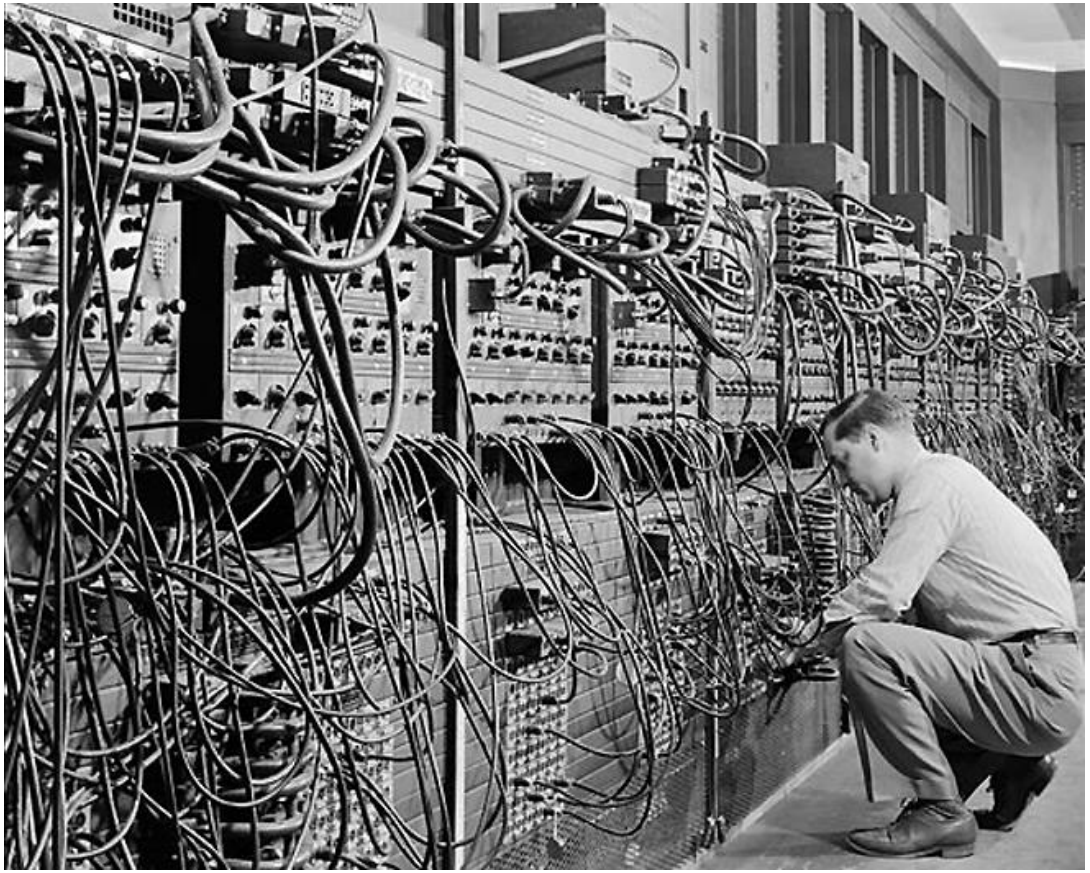
создатель первого программируемого компьютера и первого языка программирования высокого уровня

Z-серия цифровых компьютеров:

- 1941 год, компьютер Z3 - первое в мире *программно управляемое* вычислительное устройство:
  - ✔ работающее на базе *двоичной логики*;
  - ✔ способное выполнять *математические операции с плавающей запятой*;
  - ✔ способное *долговременно хранить программы* на перфорированной 35-мм пленке.
- 1942 год, первый в мире высокоуровневый язык программирования *Планкалкюль* (от нем. *Plankalkül - планирующее исчисление*)
  - ✔ средства присваивания и вызова подпрограмм;
  - ✔ условные переходы и циклы;
  - ✔ массивы и иерархические структуры данных;
  - ✔ операции с плавающей запятой;
  - ✔ обработка исключений;
  - ✔ отсутствие компилятора…

о существовании языка Планкалкюль широкой общественности стало известно лишь в 1972 году

Первые практические решения



Первый электронный компьютер общего назначения **ЭНИАК**

(ENIAC - Electronic Numerical Integrator and Computer, 1943 - 45 гг, США)

ЭНИАК - основные характеристики

- способен реализовать *любую вычислимую функцию* (так называемое, *требование полноты по Тьюрингу*);
- высокая, по тем временам, скорость вычислений (в десятичной системе счисления);
- *низкая отказоустойчивость* по причине хрупкости электровакуумных ламп, которых в схемах ЭНИАКа насчитывалось более 17 тысяч при общем весе конструкции - 27 тонн (*несколько секунд полезных вычислений чередовались с несколькими часами простоя для поиска и замены вышедших из строя ламп*);
- *монтажный способ программирования* - для каждой новой задачи необходимо полностью «перепрограммировать» компьютер путем ручного переключения тысяч тумблеров и проводных контактов;
- для обслуживания такой сложной и капризной конструкции требовалось большое количество высококвалифицированных «программистов», глубоко знакомых с аппаратным устройством компьютера, а также с принципами математики, физики, электротехники и радиоэлектроники.

Ассемблеры

Ассемблер - машинно-ориентированный язык низкого уровня, команды которого максимально приближены к командам конкретного процессора.

```
-------------------------------------------------------------------------

                        TITLE   CASE    (COM) Перекод. в заглавные буквы
0000                    CODESG  SEGMENT PARA 'CODE'
                        ASSUME  CS:CODESG,DS:CODESG,SS:CODESG
0001                    ORG     100H
0001  EB 1C 90          BEGIN:  JMP     MAIN
                        ; ---------------------------------------
0003  43 68 61 6E 67 65 TITLEX  DB      'Change to uppercase letters'
      20 74 6F 20 75 70
      70 65 72 63 61 73
      65 20 6C 65 74 74
      65 72 73
                        ; ---------------------------------------
011E                    MAIN    PROC    NEAR
011E  8D 1E 0104 R              LEA     BX,TITLEX+1  ;Адрес первого симв.
0122  B9 001F                   MOV     CX,31        ;Число символов
0125                    B20:
0125  8A 27                     MOV     AH,[BX]      ;Символ из TITLEX
0127  80 FC 61                  CMP     AH,61H       ;Это
012A  72 0A                     JB      B30          ; прописная
012C  80 FC 7A                  CMP     AH,7AH       ; буква
012F  77 05                     JA      B30          ; ?
0131  80 E4 DF                  AND     AH,11011111B ;Да - преобразовать
0134  88 27                     MOV     [BX],AH      ;Записать в TITLEX
0136                    B30:
0136  43                        INC     BX           ;Следующий символ
0137  E2 EC                     LOOP    B20        ;Повторить цикл 31 раз
0139  C3                        RET
013A                    MAIN    ENDP
013A                    CODESG  ENDS
                        END     BEGIN
-------------------------------------------------------------------------
```

Пример листинга на языке ассемблер для IBM PC x86 с соответствующими машинными кодами (слева). ***Программа для замены строчных букв на прописные***

Ассемблеры - основные характеристики

- Суть: команды процессора, представляющие собой двоичные коды (последовательности нулей и единиц), просто заменились соответствующими алфавитными мнемокодам, более легкими в запоминании и употреблении;
- каждый ассемблер строго привязан к своему типу процессора и архитектуре ЭВМ (отсюда *высокая эффективность использования ресурсов компьютера*);
- программирование на ассемблере требует от программиста глубокого знания технических и конструктивных особенностей компьютера;
- низкая читаемость программ и большой объем исходного кода;
- трудоемкость отладки;
- трудности масштабирования и переноса программ на другие типы ЭВМ;
- ассемблеры используются и по сей день *для решения низкоуровневых системных задач*: написание драйверов устройств, прошивок BIOS, программирование ядра операционных систем, создание межплатформенных модулей совместимости и т.п.

Ассемблеры - основные характеристики

Предпосылки возникновения **императивного стиля программирования**:

- Размеры памяти и быстродействие первых компьютеров были крайне невелики, а их стоимость очень высока, поэтому для программистов того времени *вопрос эффективности программ стоял на первом месте*;
- В погоне за этой эффективностью пошли по наиболее простому пути: *архитектура языков программирования должна быть максимально приближена к архитектуре компьютера*;
- Программа должна была состоять из *последовательности* процессорных инструкций, модифицирующих память;
- За последующие почти 65 лет существенно изменились методы и средства императивного программирования, однако основная идея осталась неизменной: *программа описывает процесс пошагового решения задачи в той последовательности как ее решает компьютер*;
- получаемые императивные программы очень мало напоминают первоначальную человеческую постановку задачи, в которой нет никаких регистров, массивов, подпрограмм и прочих специфических компьютерных понятий…

Первое поколение языков (1953 - 1958 гг.)

Поскольку первые программы решали всецело математические расчетные задачи, в 1953 году 28-летний сотрудник компании IBM **Джон Бэкус (John Backus)** предложил простую и замечательную идею: *создать специальную программу, способную считывать формулы в их универсальном алгебраическом виде и выполнять их независимо от типа ЭВМ*.

- **1957 год.** Под руководством Бэкуса был создан язык программирования высокого уровня **FORTRAN-1** (от англ. FORmula TRANslator – транслятор формул) и *транслятор* с этого языка. Это событие положило начало первому поколению языков высокого уровня.
- **1958 год.** Языки математических формул **ALGOL-58** (от англ. ALGOrithmic Language – алгоритмический язык) и **FLOW-MATIC** (отличался тем, что был предназначен не для обработки математических формул, а для задач обработки коммерческих данных - в основном таблиц).
- Языки математических формул получили стремительное распространение, ALGOL – в академических кругах Европы и СССР, а FORTRAN и FLOW-MATIC пользовались большей популярностью в США и Канаде.

Форма Бэкуса-Науера (БНФ, BNF)

В 1959 году Бэкус закончил разработку так называемой *нормальной формы Бэкуса* - формального способа описания синтаксиса алгоритмических языков.

Некоторое время спустя, способ формального описания языка был усовершенствован **Питером Науром**. *Форма Бэкуса-Наура* (БНФ, BNF) стала активно использоваться на этапе разработки новых языков.

БНФ-описание состоит из набора правил вида:

```
<определяемый символ> ::= <посл.1> | <посл.2> | . . . | <посл.n>
```

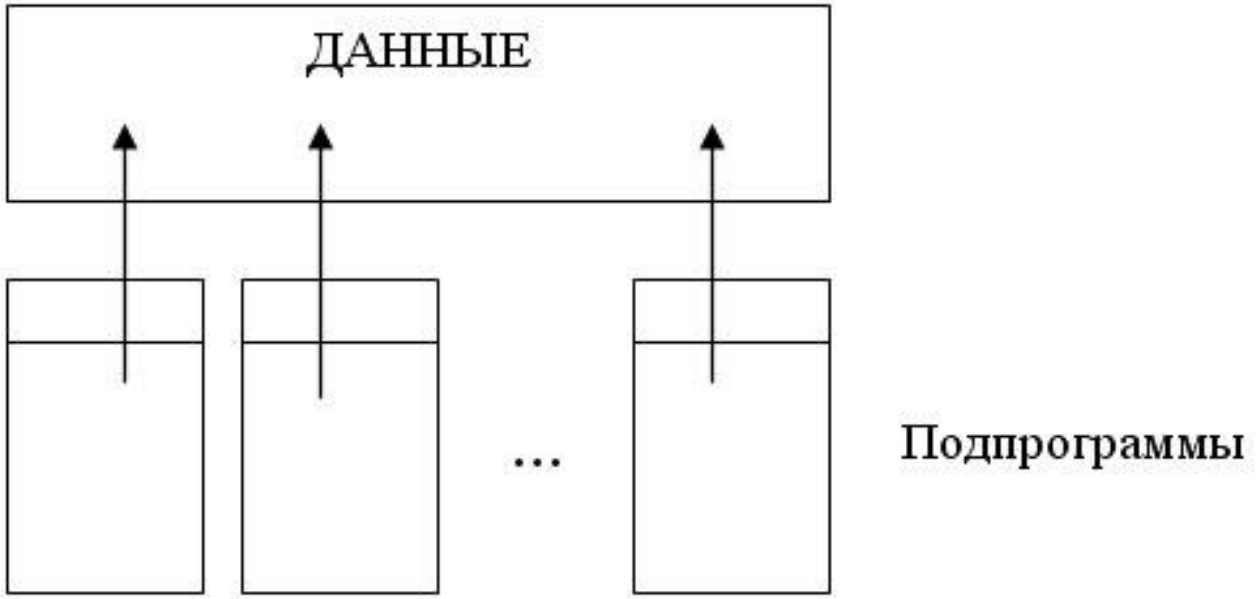
Описание оператора `if` языка PASCAL в БНФ:

```
<условный оператор if> ::= if <булево выражение> then <оператор> [else <оператор>]
<булево выражение> ::= "NOT" <булево выражение>
    | <булево выражение> <логическая операция> <булево выражение>
    | <выражение> <операция сравнения> <выражение>
<логическая операция> ::= "OR" | "AND"
<выражение> ::= <переменная> | <строка> | <символ>
<операция сравнения> ::= "=" | "<" | ">"
 …
```

Возникновение БНФ послужило важнейшим фактором в дальнейшей эволюции языков программирования.

Второе поколение языков (1959 - 1961 гг.)

Важнейшими элементами конструкции программ второго поколения выступают **подпрограммы**, которые имеют доступ к общей (глобальной) памяти, выделяемой под данные:



Второе поколение языков (1959 - 1961 гг.)

Язык программирования	Характеристики
FORTRAN-II	поддержка процедурного программирования (появилась возможность писать подпрограммы и функции); раздельная компиляция частей программы.
ALGOL-60	блочная структура программы, состоящая из чётко описанных и отделённых друг от друга частей (при помощи ключевых слов begin и end); типизация данных; условное или циклическое исполнение вложенных наборов операторов, что позволило отказаться от использования безусловных переходов; рекурсивный вызов процедур; передача параметров в процедуры.

Второе поколение языков (1959 - 1961 гг.)

Язык программирования	Характеристики
COBOL (от англ. COmmon Business-Oriented Language – общий бизнес-ориентированный язык)	разработан, прежде всего, для решения задач в экономической сфере (обработка заказов, ведение бухгалтерии, планирование производства); машинно-независимость и максимальная приближенность к естественному английскому языку (что положило основу самодокументируемым программам); типизация данных (основные типы данных: записи, файлы, таблицы и списки); работа с файлами; работа с подпрограммами (параграфами); способен лишь на простейшие алгебраические вычисления, для сложных инженерных расчетов этот язык не годится.

Второе поколение языков (1959 - 1961 гг.)

Язык программирования	Характеристики
LISP (от англ. LISt Processing — обработка списков)	▯ представление программы в виде *системы линейных списков символов, которые являются основным типом данных языка*; ▯ обработка списков: в виде списков удобно представлять алгебраические выражения, графы, элементы конечных групп, множества, правила вывода и многие другие сложные объекты; ▯ *функциональная направленность*, т. е. программирование ведется с помощью функций (функция понимается как правило, сопоставляющее элементам некоторого класса соответствующие элементы другого класса); ▯ впервые использован механизм *указателей*; ▯ рекурсивный вызов функций; ▯ предназначен для составления программ, цель которых не носит численного характера, удобен для решения задач в области искусственного интеллекта и обработки символьной информации. Появилось понятие *символьных вычислений.*

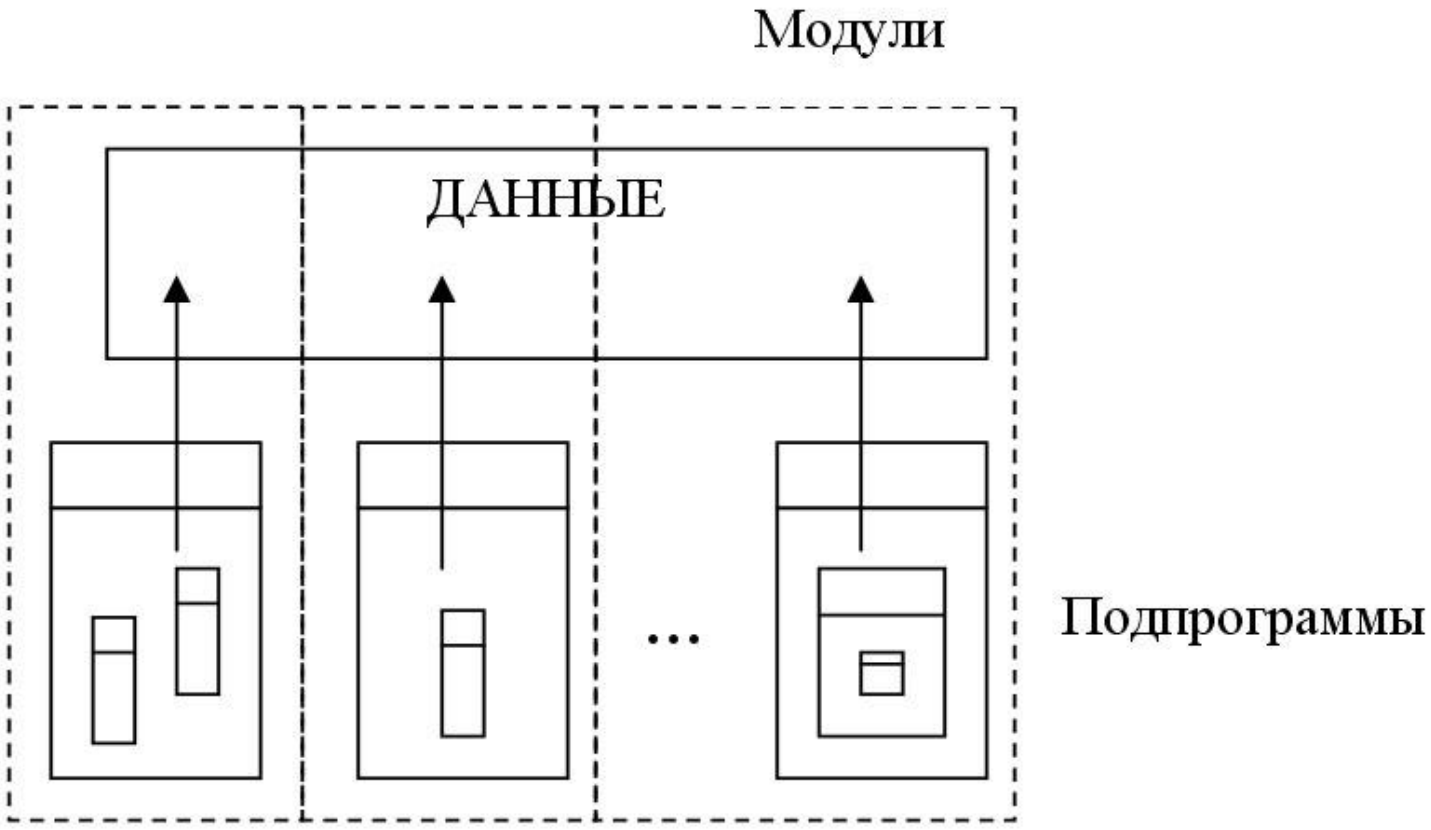
Второе поколение языков (1959 - 1961 гг.)

Общие выводы:

- Несмотря на существенный прогресс в области структуризации кода и типизации данных, программы второго поколения оставались *крайне тяжелыми в отладке*.
- выделяемая область памяти была общей для всех частей программы, что являлось причиной *большого количества наведенных ошибок*;
- *большое количество перекрестных связей между подпрограммами, низкоуровневость описания данных, беспорядочность потоков управления* – все это значительно снижало надежность программы и достоверность ее результатов.

Третье поколение языков (1962 - 1970 гг.)

В языках третьего поколения продолжается совершенствование *структурного подхода* к разработке ПО. Появляются механизмы вложения подпрограмм *с ограничением области видимости компонентов*, программы приобрели ярко выраженную модульную структуру.



Третье поколение языков (1962 - 1970 гг.)

Язык программирования	Характеристики
PL/1 (от англ. Programming Language 1 – язык программирования номер 1)	разработан в 1964 фирмой IBM на основе языков FORTRAN, ALGOL, COBOL – такое объединение позволило решать научные, инженерные и экономические задачи средствами одного универсального языка программирования; развитая система встроенных типов данных и неявные способы преобразования между ними; несколько способов динамического выделения памяти; поддержка на уровне языка мультизадачности и асинхронного ввода-вывода; наличие мощного препроцессора; гибкость и вариативность языковых конструкций, что с одной стороны приводит к более компактному коду, а с другой – затрудняет чтение программы и изучение самого языка; по возможностям язык сильно опережал свое время и, как следствие, содержал множество маловостребованных и сложных конструкций – с этим было связано отсутствие надежных компиляторов, поддерживающих все возможности языка; неоптимальность скомпилированного кода, что было принципиальным недостатком для программирования математических расчётных задач.

Третье поколение языков (1962 - 1970 гг.)

Язык программирования	Характеристики
ALGOL-68	• развитие языка ALGOL-60 в период 1964 – 1968 гг.; • средства организации параллельных вычислений; • операции со структурами как с цельными объектами; • встроенная реализация матричных операций, что давало особые удобства для инженеров и проектировщиков; • возможность переопределения синтаксиса языка и операторов, что давало возможность реализации алгоритмов любого уровня абстракции и написания программ практически в любом стиле; • мультиязычность за счет использования таблиц трансляции, позволяющая для каждого естественного языка определить свой набор ключевых слов Алгола-68; • сложность синтаксиса и читаемости программ; • отсутствие надежных компиляторов, поддерживающих все возможности языка.

Третье поколение языков (1962 - 1970 гг.)

Язык программирования	Характеристики
PASCAL (назван в честь французского математика, физика и философа Блеза Паскаля)	разработан *Никлаусом Виртом* в 1970 году как развитие языка ALGOL-60; строгая и безопасная типизация; простота и строгость синтаксиса, сведены к минимуму все возможные синтаксические неоднозначности; развитая система встроенных типов данных: записи, массивы, файлы, множества, указатели и др.; основные недостатки: отсутствие нормальных средств работы с динамической памятью, ограниченная библиотека ввода-вывода, отсутствие средств для подключения функций написанных на других языках, отсутствие средств раздельной компиляции; благодаря своей простоте и наглядности, язык обрел огромную популярность и пользуется ею по сей день – реализовано множество компиляторов и диалектов Паскаля.

Третье поколение языков (1962 - 1970 гг.)

Язык программирования	Характеристики
SIMULA 67	впервые описан в 1970 году сотрудниками норвежскими учеными Уле-Йоханом Далем (Ole-Johan Dahl) и Кристеном Нюгардом (Kristen Nygaard); первый *объектно-ориентированный язык*, основное назначение – разработка и моделирования сложных систем; развитие языков SIMULA 1 и ALGOL 60 за счет добавления таких объектно-ориентированных свойств, как *классы, инкапсуляция и наследование*; *абстрактное представление данных*, позволяющих естественным образом описывать специфику конкретной предметной области.

Третье поколение языков (1962 - 1970 гг.)

Основные выводы:

В своей завершающей стадии, развитие языков программирования третьего поколения способствовало созданию больших проектов, подразумевающих *параллельную работу групп программистов над отдельными частями общей системы*.

Четвертое поколение. Языки ООП (1970 - наши дни)

- Введено два новых понятия: **класс** и **объект**. Под классом понимается пользовательский абстрактный тип данных, объединяющий в себе и данные и функции (*методы* для обработки этих данных). Объект является *экземпляром* класса, т.е. элементом данных, типом которых выступает данный класс.
- **Объектно-ориентированный подход (ООП)** является дальнейшим развитием парадигмы процедурного программирования, отличающийся главным образом тем, что основным элементов конструкции программы служит не подпрограмма, а модуль – независимая часть кода, составленная из связанных классов и объектов.
- Классы ограничивают видимость своих данных, предоставляя к ним доступ только свои собственным функциям-методам или специально определенным дружественным функциям. Такое свойство получило название *инкапсуляции*.
- Кроме этого, классы могут *наследовать* свойства и права доступа к данным от других (родительских) классов, выстраивая, таким образом, сложные иерархические конструкции абстрактных типов данных.
- важным свойством ООП выступает свойство *полиморфизма* в двух его разновидностях: *параметрического* и *ситуационного*. Параметрический полиморфизм подразумевает возможность создания обобщенных функций, в которых значения обрабатываются схожим образом вне зависимости от их типа. Ситуационный (ad hoc) полиморфизм предоставляет возможность иметь несколько функций с одинаковым названием (полиморфных функций), но вызываемых с разным набором аргументов. *Таким образом, с внешней точки зрения, функция может демонстрировать разное поведение в зависимости от типа своих параметров.*

Четвертое поколение. Языки ООП (1970 - наши дни)

- Общая координация и управления системой объектов осуществляется за счет *событий* - специальных программных сообщений о действиях пользователя (например, нажатие кнопки мыши в определенном участке экрана), о поведении других программ, о состоянии операционной системы и др.
- В ООП программа больше не является последовательностью операторов, а представляет собой набор объектов, ожидающих «своих» событий и заданным образом реагирующих на них (*событийное управление вычислениями*).
- ООП оказал решающее влияние на технологии разработки больших проектов и систем, в рамках которых координируется работа сотен и тысяч программистов. *Создание отдельных классов, их модульная компоновка, отладка, тестирование и проектирование системы в целом могут производится распределено разными группами разработчиков* даже без непосредственного контакта друг с другом.

Четвертое поколение. Языки ООП (1970 - наши дни)

Язык программирования	Характеристики
SMALLTALK-80 *Аналогичными свойствами обладает современный ООП-язык Java (кроме динамической типизации)*	*всякая конструкция языка является объектом* – даже такие конструкции как условные операторы и циклы не являются частью языка, а реализуются при помощи объектов (например, специальный логический объект, получая сообщение, принимает решение о дальнейшей передаче управления); выполнение программы состоит из отправки сообщений между объектами; *динамическая типизация*, согласно которой программист не указывает типы переменных в программе – корректность используемых типов контролируется самими объектами; программист не заботится о сборке «мусора», сборщик встроен в язык; программы компилируются в машинно-независимый код низкого уровня (байткод) и обычно исполняются виртуальной машиной (что обеспечивает определенную аппаратную независимость программ).

Четвертое поколение. Языки ООП (1970 - наши дни)

Язык программирования	Характеристики
Object Pascal (разработка фирмы Apple, 1986 г.)	• перегрузка процедур, функций и операторов не входящих в состав каких-либо классов; • динамическая типизация при которой *переменная связывается с типом в момент присваивания значения, а не в момент ее объявления* (например, был введен вариантный тип данных, который не известен на этапе компиляции и может изменяться по ходу выполнения программы); • работа только с *динамическими экземплярами классов* (статические экземпляры создаются принудительно в момент компиляции, динамические экземпляры и память для них выделяются исходя из логики программы - по аналогии с динамическими массивами); • появление понятия *визуального объекта* (заложены основы визуального подхода к программированию).

Четвертое поколение. Языки ООП (1970 - наши дни)

Язык программирования	Характеристики
C++ (разработка языка началась Бьерном Страуструпом в 1979 году и доведена до коммерческой реализация в 1985 году)	• совмещение различных парадигм программирования (процедурной, объектно-ориентированной); • поддержка всех четырех свойств ООП — абстракция данных, инкапсуляция, множественное наследование и полиморфизм; • поддержка парадигмы *обобщенного программирования* за счет использования шаблонов функций и классов; • полная совместимость с библиотеками языка С; • перегрузка функций и операторов; • совмещение статический и динамической типизации данных; • три уровня защиты данных при наследовании: публичный, защищённый и закрытый.

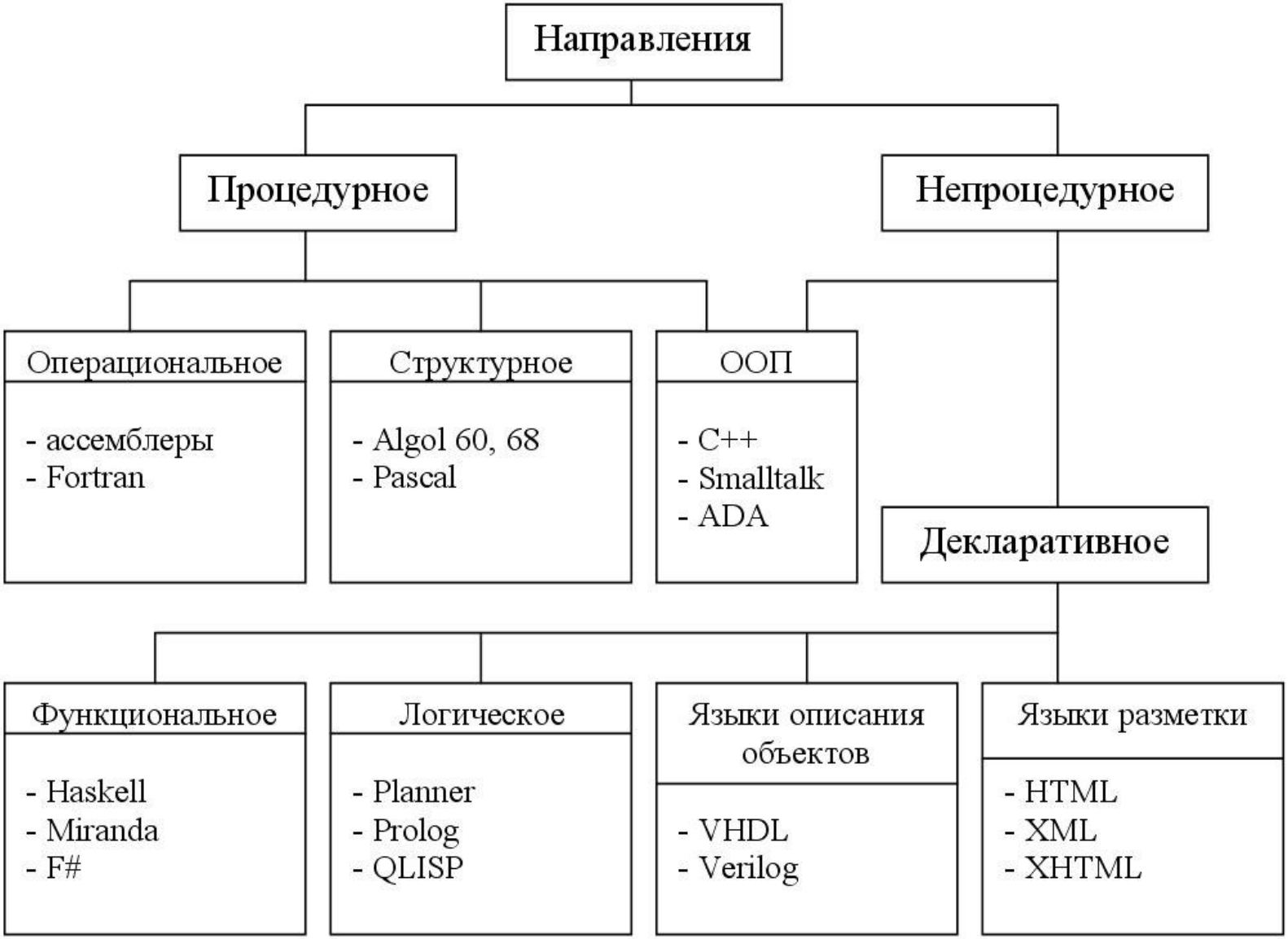
Четвертое поколение. Языки ООП (1970 - наши дни)

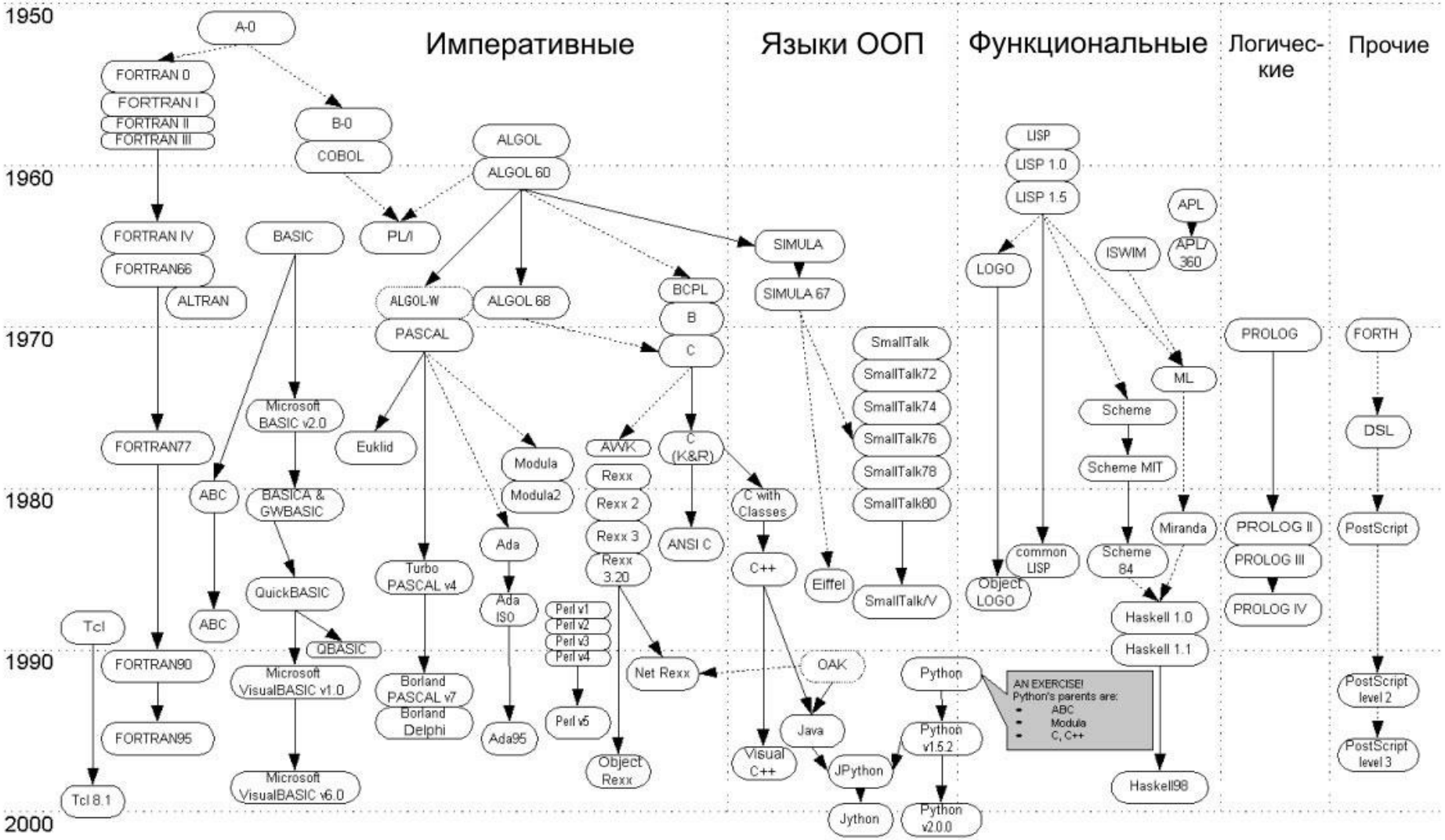
Язык программирования	Характеристики
CLOS (от англ. Common Lisp Object System – объектная система общего ЛИСПа)	• интеграция объектно-ориентированной и функционально-логической парадигм программирования, объектно-ориентированное развитие стандарта языка Common Lisp; • использование механизма *множественной диспетчеризации*, согласно которому выбор одной из нескольких одноименных функций происходит в зависимости от динамических типов или значений аргументов; • методы не определяются внутри классов, а специальным образом группируются в «обобщённые функции»; • динамическая типизация при которой не только содержимое, но и структура объектов может меняться во время работы программы; • множественное наследование.

Четвертое поколение. Языки ООП (1970 - наши дни)

Язык программирования	Характеристики
ADA (создан в период 1979 – 1980 гг. по заказу министерства обороны США)	• синтаксис языка унаследован от таких языков как Pascal и Algol – первые реализации языка не поддерживали ООП и соответствовали процедурной парадигме программирования (средства поддержки ООП добавлены в язык в 1995 году); • *строгая типизация*, при которой не допускается работа с объектами, не имеющими типов, автоматические преобразования типов допускаются в крайне редких случаях; • мощные средства создания пользовательских типов данных; • мощные средства обработки исключений; • *возможность передачи фактических параметров в произвольном порядке* с указанием имен формальных параметров; • возможность перегрузки процедур, функций и операторов; • развитые *средства поддержки параллельного программирования*, встроенные непосредственно в язык;

Направления развития языков программирования





Генеалогическое дерево языков программирования до 2000 года