# Введение в языки программирования

## Язык программирования

Язык программирования — это формальный язык, предназначенный для записи компьютерных программ. Формальность языка заключается в том, что каждый язык содержит конечное множество лексических конструкций.

Язык программирования в первую очередь отличается от обычного, тем, что целью «общения» является вычислительная система. Языки программирования используют специальные конструкции для определения и манипулирования структурами данных и управления процессом вычислений.

В идеале язык программирования стандартизируется — определяется его спецификация, включающая семантику и синтаксис языка.

Однако, на практике стоит учитывать еще и конкретные реализации — программные средства в виде программ трансляторов — компиляторов или интерпретаторов (а также их гибридными вариантами). Такие реализации можно различать по производителю (айти компании), по марке и варианту (обычно это выражается в указании версии программы), времени выпуска, насколько полно был реализован стандарт и другим признакам.

## История языков программирования

### Предтечи.

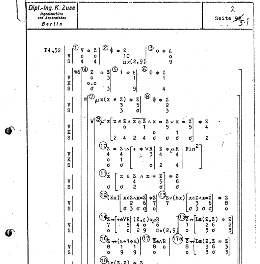
Пожалуй, первым языком программирования, как языке записи алгоритмов, для последующего выполнения на вычислительной машине, можно считать псевдомашинный алгоритмический код Ады Лавлейс, для разностной машины Беббиджа.

Ада Лавлейс (1815—1852), дочь известнейшего английского поэта лорда Байрона. Получила математической образование. Была хорошо знакома с учеными и выдающимися деятелями своего времени: Чарльзом Бэббиджем, Майклом Фарадеем, Дэвидом Брюстером, Чарльзом Уитстоном, Чарльзом Диккенсом и другими.

Один из алгоритмов, описанных ею — алгоритм вычисления чисел Бернулли, был признан как специально написанная программа для исполнения на вычислительной машине (разностной машине Беббиджа). Несмотря на то, что сама машина при жизни изобретателя и Ады не была построена, эта программа признана первой в мире, а сама Ада Лавлейс считается первым программистом.

В своих записях она рассуждала о том, что, подобно тому, как Жаккардов ткацкий станок может ткать цветы и листья, аналитическая машина способна создавать алгебраические формулы, а в перспективе — писать музыку, писать картины — и укажет «науке такие пути, какие нам и не снились».

В 40 годы, в Германии, инженер Конрад Цузе спроектировал и построил вычислительную машину Z4, параллельно с созданием машины, он разрабатывал первый в мире высокоуровневый язык программирования, созданный. В переводе на русский это название соответствует выражению «планируемые вычисления» или «запланированные вычисления», то есть «программирование».

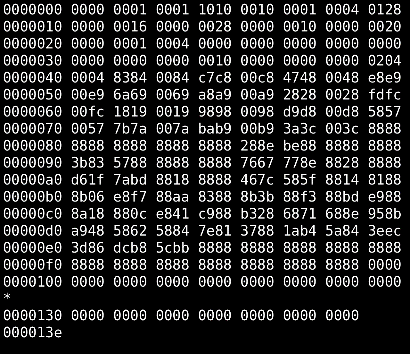
Планкалкюль поддерживал операции назначения, вызов подпрограмм, условные операторы, итерационные циклы, арифметику с плавающей запятой, массивы, иерархические структуры данных, утверждения, обработку исключений и многие другие вполне современные средства языков программирования.

### Машинный язык

Электронно-вычислительная машина, ЭВМ, компьютер — все это одно название современных вычислительных устройств. Физические принципы работы основаны на хранении и передачи электрических напряжений, а значит имеющих возможность передать только один бит информации.

1 бит информации — символ или сигнал, который может принимать два значения: включено или выключено, да или нет, высокий или низкий, заряженный или незаряженный; в двоичной системе исчисления это 1 (единица) или 0 (ноль). Это минимальное количество информации, которое необходимо для ликвидации минимальной неопределенности.

Этот набор битов информации, то есть набор нулей и единиц и есть машинный язык — единственный который «понимает» компьютер на базовом уровне.

Первые годы развития ЭВМ, человеку было необходимо составлять программы на языке, понятном компьютеру, то есть в машинных кодах. Каждая команда состояла

* из кода операций
* адресов операндов,

выраженных в виде различных сочетаний единиц и нулей.

Итак, любая программа для процессора выглядела на то время как последовательность единиц и нулей.

На практике, для постоянного общения с компьютером, такой язык громоздок и неудобен. При пользовании им легко допустить ошибку, записав не в той последовательности 1 или 0. Программу очень трудно контролировать. Кроме того, при программировании в машинных кодах надо хорошо знать внутреннюю структуру ЭВМ, принцип работы каждого блока. И самое плохое в таком языке, что программы на данном языке — очень длинные последовательности единиц и нулей являются машинно-зависимыми, то есть для каждой ЭВМ необходимо было составлять свою программу, а также программирование в машинных кодах требует от программиста много времени, труда, повышенного внимания.

### Ассемблер

Уже в начале 50-х годов были разработаны мнемонические языки для описания машинных команд более-менее удобными для восприятия словесными сокращениями. Например, вместо набора нулей и единиц 00101100, которые могли обозначать сложение двух регистров процессора, появилась возможность написать add.

Ассемблер — язык программирования низкого уровня. Язык программирования низкого уровня — язык программирования, который ориентирован на конкретный тип процессора и учитывает его особенности. В данном случае «низкий уровень» не значит «плохой». Имеется в виду, что операторы языка близки к машинному коду и ориентированы на конкретные команды процессора. Появление языка ассемблера значительно облегчило жизнь программистов, так как теперь вместо рябящих в глазах нулей и единиц, они могли писать программу командами, состоящими из символов приближенных к обычному языку. Для того времени этот язык был новшеством и пользовался популярностью так как позволял писать программы небольшого размера, что при тех машинах критерий значительный.

Программа Hello World на assembler Intel:

; Hello World for Intel Assembler (MSDOS)

mov ax,cs

mov ds,ax

mov ah,9

mov dx, offset Hello

int 21h

xor ax,ax

int 21h

Hello:

db "Hello World!",13,10,"$"

Вместе с тем, отметим, что нет как такого одного Ассемблера. Дело в том, что ассемблер это по сути «очеловеченный» машинный язык, а значит, что для каждого типа процессора существует свой Ассемблер. Конечно, у них есть общие команды и принципы, но все-таки есть и различия.

Также надо понимать, что программирование на языке ассемблера, хоть и стало понятнее, однако, по-прежнему оставалось очень трудоемким, ведь, по сути, приходилось вручную описывать каждый шаг процессора. Все это привело к созданию языков третьего поколения — языков высокого уровня.

Программа Hello World на assembler ARM:

/\* Hello world in ARM assembly (Android devices) \*/

.data

msg:

.ascii "Hello, World!\n"

len = . - msg

.text

.globl \_start

\_start:

mov %r0, $1

ldr %r1, =msg

ldr %r2, =len

mov %r7, $4

swi $0

mov %r0, $0

mov %r7, $1

swi $0

Но сегодня применение ассемблера не прекратилось. Он используется в написании отдельных фрагментов программ или иногда в написании самих программ. Примеров может быть много, но самые яркие это использование ассемблера в написании драйверов, игровых движков и загрузчиков ОС (операционных систем).

Также ассемблером пользуются хаккеры, для написания вирусов, шпионских программ (троянов), различных «отмычек». Это связанно с тем, что объем кода таких программ гораздо меньше, а значит и скорость работы значительно выше скорости программы, написанной на языке программирования высокого уровня.

Программа Hello World на assembler Win32:

; Hello world in Assembler for the Win32 architecture

TITLE Hello world in win32. Tasm

VERSION T310

Model use32 Flat,StdCall

start\_code segment byte public 'code' use32

begin:

Call MessageBox, 0, offset sHallo, offset caption, 0

Call ExitProcess, 0

start\_code Ends

start\_data segment byte public 'data' use32

sHallo db 'Hello world',0

caption db "Hi",0

start\_data Ends

End begin

По этой же причине (малый объем и быстродействие) и разработчики антивирусов так же используют ассемблер в некоторых модулях своих программ.

Важно заметить, что термин Ассемблер относится не только к языкам программирования, но и к специальным программам, которые переводят программу с языка Ассемблера и создают — «собирают» (assembly) исполняемую программу на машинном языке.

### Языки третьего поколения (языки высокого уровня)

В 50-60 годы возникло понимание того, что привязка языка программирования к определенной архитектуре процессора (иными словами — к одной платформе), тормозит развитие программирования. Стали разрабатываться и внедряться языки высокого уровня, которые уже были кроссплатформенными в плане синтаксиса. То есть текст программы на таком языке одинаково подходил для любой платформы. Другое дело, что такой текст должен был быть «переведен» — оттранслирован — в машинный код конкретной архитектуры процессора и операционной системы. Это осуществляли специальные программы трансляторы (компиляторы и интерпретаторы).

#### FORTRAN

Первым языком высокого уровня признан язык Фортран (FORTRAN — **Form**ula **Tran**slation). Созданный в 1954-57 годах специалистами компании IBM под руководством Джона он стал практически сразу же популярен среди ученых и инженеров для написания программ с интенсивными вычислениями. Включение типа данных «комплексное число», сделало его особенно удобным для технических приложений. В Советском Союзе в конце 60-х готов был написан собственный компилятор для языка Фортран.

Программа Hello World на Fortran:

C Hello World in Fortran

PROGRAM HELLO

WRITE (\*,100)

STOP

100 FORMAT (' Hello World! ' /)

END

Фортран широко используется и сегодня. В основном для научных и инженерных вычислений. Он прекрасно подходит для решения численных задач, так как за время его существования было написано множество библиотек. Среди учёных ходит такая присказка, что любая математическая задача уже имеет решение на Фортране, и, что недалеко от истины: среди тысяч фортрановских пакетов и библиотек, можно найти и пакеты для работы с матрицами, и пакет для решения сложных интегральных уравнений, и многие другие.

Программа Hello World на Fortran90:

! Hello World in Fortran 90 and 95

PROGRAM HelloWorld

WRITE(\*,\*) "Hello World!"

END PROGRAM

#### ALGOL

Фортран создавался в США, и в Европе небезосновательно считали, что IBM начнет в качестве монополиста диктовать условия. В 1958 году был создан первый стандарт языка Algol (algorithmic language). Вскоре его доусовершенствовали, в результате появился язык Алгол-60.

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 главный блок, а также описания подпрограмм. 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.

Программа Hello World на Algol60:

'BEGIN'

'COMMENT' Hello World in Algol 60;

OUTPUT(4,'(''('Hello World!')',/')')

'END'

Были выделены структурные управляющие конструкции: ветвления, циклы, последовательные участки, исполняющие условно или многократно вложенные наборы операторов. То, что сегодня кажется очевидным, в те годы было настоящим прорывом.

#### LISP

LISP является аббревиатурой выражения LISt Processing — обработка списков. Идея языка была предложена Дж. Маккарти в 1960 году, и сводилось к тому, что обработка была направлена не на вычисление, а на обработку информации, оформленную в виде списков.

Программа Hello World на Lisp:

;;; Hello World in Emacs Lisp.

(defun hello-world()

"Display the string hello world."

(interactive)

(message "hello world"))

В виде списков удобно представлять алгебраические выражения, графы, элементы конечных групп, множества, правила вывода и многие другие сложные объекты. Списки являются наиболее гибкой формой представления информации в памяти компьютеров. Неудивительно, что удобный язык Лисп, специально предназначенный для обработки списков, быстро завоевал популярность.

#### COBOL

COBOL (**co**mmon **b**usiness-**o**riented **l**anguage) — был разработан в 1959 году и предназначался прежде всего для написания программ для разработки бизнес приложений, а также для работы в экономической сфере.

Программа Hello World на Lisp:

\* Hello World in COBOL

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

IDENTIFICATION DIVISION.

PROGRAM-ID. HELLO.

ENVIRONMENT DIVISION.

DATA DIVISION.

PROCEDURE DIVISION.

MAIN SECTION.

DISPLAY "Hello World!"

STOP RUN.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Создатели языка ставили своей целью сделать его машинонезависимым и максимально приближенным к естественному английскому языку. Обе цели были успешно достигнуты; программы на COBOL считаются понятными даже неспециалистам, поскольку тексты на этом языке программирования не нуждаются в каких-либо специальных комментариях (самодокументирующиеся программы).

#### BASIC

Бейсик был придуман в 1964 году преподавателями Дартмутского Колледжа Джоном Кемени и Томасом Курцем, и под их руководством был реализован командой студентов колледжа. Со временем, когда стали появляться другие диалекты, изначальную реализацию стали называть Dartmouth BASIC.

Программа Hello World на Basic:

10 REM Hello World in BASIC

20 PRINT "Hello World!"

Бейсик был спроектирован так, чтобы студенты могли без затруднений писать программы, используя терминалы с разделением времени. Он предназначался для более «простых» пользователей, не столько заинтересованных в скорости исполнения программ, сколько просто в возможности использовать компьютер для решения своих задач, не имея специальной подготовки.

Бейсик в полной мере испытал на себе такое хорошо известное в 1970—1980 явление, как массовое размножение версий и диалектов языка высокого уровня, в значительной степени несовместимых между собой. Причин этому было несколько. Поскольку язык не рассматривался как промышленный, никаких специальных усилий по его стандартизации не прилагалось, и для каждого компьютера создавалась своя реализация.

Периодом наибольшего расцвета и развития исходного Бейсика можно считать конец 1970-х — первую половину 1980-х годов. В этот период практически все персональные компьютеры имели собственный интерпретатор Бейсика, зачастую зашиваемый в ПЗУ, который в простейших компьютерах нередко выполнял также функции операционной системы.

Про Бейсик можно было бы написать трактат, как из малоизвестного интерпретатора он поднялся до невероятных высот популярности и после стал «притчей во языцех» как несовершенный и неудобный язык. Однако справедливости ради отметим, что это была прекрасная история, а главное современное перерождение в Visual Basic .NET подарило ему вторую жизнь (хотя конечно новая версия практически не имеет ничего общего с классическим Basic)

### Структурное программирование и процедурные языки

В конце 60-х начале 70-х годов XX века, на фоне роста сложности программ, стали активно разрабатываться новые технологические подходы программирования. Требовалось повысить эффективность и производительность работы коллективов программистов. Так постепенно выкристаллизовалось

**Структурное программирование** — парадигма программирования, в основе которой лежит представление программы в виде иерархической структуры блоков. Концептуализирована на фундаменте теоремы Бёма-Якопини, математически обосновывающей возможность структурной организации программ, и работы Эдсгера Дейкстры «О вреде оператора goto».

В соответствии с парадигмой, любая программа, которая строится без использования оператора goto, состоит из трёх базовых управляющих конструкций: последовательность, ветвление, цикл; кроме того, используются подпрограммы. При этом разработка программы ведётся пошагово, методом «сверху вниз».

Вследствие этого потребовалось доработать существующие языки или создать новые.

#### PL/1

ПЛ/1 (Programming Language I — «язык программирования номер один») один из первых универсальных языков на котором возможны разработки вычислительных, научных, инженерных и даже бизнес программных систем. Содержит настолько широкий набор синтаксических конструкций и встроенных функций, что, пожалуй, не существует ни одного компилятора, поддерживающего все предусмотренные в спецификации языка возможности. Поддерживает рекурсию и структурное программирование.

Программа Hello World на PL/1:

/\* Hello World in PL/1 \*/

Hello: procedure options(main);

put skip list('Hello World!');

end Hello;

#### C

В 1973 году сотрудником компании Bell Labs Дениссом Ритчи, был представлен разработанный им язык программирования С (Си). Название языка свидетельствовало, что язык был прямым продолжением и наследником языка B.

Программа Hello World на С:

#include <stdio.h>

int main()

{

printf("Hello, World!\n");

return 0;

}

Первоначально был разработан для реализации операционной системы UNIX, но впоследствии был перенесён на множество других платформ. Согласно дизайну языка, его конструкции близко сопоставляются типичным машинным инструкциям, благодаря чему он нашёл применение в проектах, для которых был свойственен язык ассемблера, в том числе как в операционных системах, так и в различном прикладном программном обеспечении для множества устройств — от суперкомпьютеров до встраиваемых систем. Язык программирования Си оказал существенное влияние на развитие индустрии программного обеспечения, а его синтаксис стал основой для множества языков программирования.

Язык Си разрабатывался как язык системного программирования, для которого можно создать однопроходный компилятор. Стандартная библиотека также невелика. Как следствие данных факторов — компиляторы разрабатываются сравнительно легко. Поэтому данный язык доступен на самых различных платформах. К тому же, несмотря на свою низкоуровневую природу, язык ориентирован на переносимость. Программы, соответствующие стандарту языка, могут компилироваться под различные архитектуры компьютеров.

Целью языка было облегчение написания больших программ с минимизацией ошибок по сравнению с ассемблером, следуя принципам процедурного программирования, но избегая всего, что может привести к дополнительным накладным расходам, специфичным для языков высокого уровня.

#### Forth

Форт — один из первых конкатенативных языков программирования, в котором программы записываются последовательностью лексем («слов» в терминологии языка Форт). Математические выражения представляются постфиксной записью при использовании стековой нотации. Поддерживает механизмы метарасширения семантики и синтаксиса языка для адаптации к нужной предметной области. Синтаксис базового уровня в Форте прост и состоит из единственного правила: «все определения разделяются пробелами». Определения Форта могут иметь любое сочетание символов.

Программа Hello World на Forth:

' Hello world in Forth

." Hello World" CR

Ряд свойств, а именно интерактивность, гибкость и простота разработки делают Форт эффективным языком в прикладных исследованиях и при создании инструментальных средств. Очевидными областями применения этого языка являются встраиваемые системы управления. За счёт простоты транслятор, а зачастую и компилятор Форта легко реализуется для подавляющего числа микроконтроллеров, также используется как кросс-средство программирования. Применяется также при программировании компьютеров под управлением различных операционных систем или как самостоятельная операционная среда.

#### Ada

Ада — язык программирования, созданный в 1979—1980 годах в ходе проекта Министерства обороны США с целью разработать единый язык программирования для встроенных систем (то есть систем управления автоматизированными комплексами, функционирующими в реальном времени). Имелись в виду прежде всего бортовые системы управления военными объектами (кораблями, самолётами, танками, ракетами, снарядами и т. п.). Перед разработчиками не стояло задачи создать универсальный язык, поэтому решения, принятые авторами Ады, нужно воспринимать в контексте особенностей выбранной предметной области. Язык назван в честь Ады Лавлейс.

Программа Hello World на Ada:

-- Hello World in Ada

with Text\_IO;

procedure Hello\_World is

begin

Text\_IO.Put\_Line("Hello World!");

end Hello\_World;

В исходном варианте, стандартизованном в 1983 году, Ада — это структурный, модульный язык программирования, содержащий высокоуровневые средства программирования параллельных процессов. Ада — язык со строгой типизацией, в нём исключена работа с объектами, не имеющими типов, а автоматические преобразования типов сведены к абсолютному минимуму: допускается неявное приведение значения общего целого или вещественного числового типа к совместимому числовому типу.

#### Pascal

Язык программирования Паскаль был создан в 1970 году на основе языка Аогол. Язык создавался Никлаусом Виртом в конце 60-х годов после его участия в работе комитета разработки стандарта языка Алгол-68. Язык назван в честь французского математика, физика, литератора и философа Блеза Паскаля, который создал одну из первых в мире механических машин, складывающую два числа. По мысли автора языка Вирта, целью его создания указывал построение небольшого и эффективного языка, способствующего хорошему стилю программирования, использующему структурное программирование и структурированные данные.

Программа Hello World на Pascal:

program HelloWorld(output);

begin

WriteLn('Hello World!');

end.

Несмотря на то, что язык создавался исключительно как учебный, Паскаль оказался настолько понятным, удобным и легким для обучения, что стал набирать популярность как язык для промышленного программирования. Было создано множество реализаций, среди которых выделяется Turbo Pascal от компании Borland, дальнейшее развитие которой перешло в среду разработки и язык Delphi.

Следующей ступенью развития структурного программирования стала объектно-ориентированная парадигма программирования.

### Объектно-ориентированное программирование

Объектно-ориентированное программирование возникло в результате развития идеологии процедурного программирования, где данные и подпрограммы (процедуры, функции) их обработки формально не связаны.

В центре ООП находится понятие объекта. Объект — это сущность, которой можно посылать сообщения, и которая может на них реагировать, используя свои данные. Объект — это экземпляр класса. Данные объекта скрыты от остальной программы. Появление в ООП отдельного понятия класса закономерно вытекает из желания иметь множество объектов со сходным поведением. Класс в ООП — это в чистом виде абстрактный тип данных, создаваемый программистом.

Объектно-ориентированный язык программирования (ОО-язык) — язык, построенный на принципах объектно-ориентированного программирования.

Со всеми концепциями и понятиями объектно-ориентированного программирования вы познакомитесь в свое время.

#### Simula

Симула — язык программирования общего назначения, разработанный в конце 1960-х годов сотрудниками Норвежского вычислительного центра Кристеном Нюгором и Оле-Йоханом Далем для моделирования сложных систем.

Программа Hello World на Simula:

! Hello World in Simula;

BEGIN

OutText("Hello World!");

OutImage;

END

Стал первым языком со встроенной поддержкой основных синтаксических соглашений, принятых в современных языках объектно-ориентированного программирования. Считается, что язык в значительной степени опередил время, и программисты 1960-х годов оказались не готовы воспринять ценности языка, в связи с чем он не выдержал конкуренции с другими языками программирования. Прохладному отношению к языку способствовало и то обстоятельство, что его реализация была весьма неэффективна, не в последнюю очередь из-за использования сборки мусора.

Концепции ООП получили дальнейшее развитие в языке Smalltalk.

#### Smalltalk

Smalltalk — объектно-ориентированный язык программирования с динамической типизацией, основанный на идее посылки сообщений, разработанный в Xerox PARC Аланом Кэем, Дэном Ингаллсом, Тедом Кэглером, Адель Голдберг, и другими в 1970-х годах. Представляет собой интегрированную среду разработки и исполнения, объекты которой доступны для модификации через неё саму, и программирование в которой в итоге сводится к модификации её собственного поведения. Язык был представлен как Smalltalk-80.

Программа Hello World на Smalltalk:

"Hello World in Smalltalk (simple version)"

Transcript show: 'Hello World!'.

Smalltalk является одним из многих объектно-ориентированных языков, основанных на языке Симула, который сам оказал большое влияние на развитие многих объектно-ориентированных языков. Многие идеи 1980-х и 1990-х по написанию программ появились в сообществе Smalltalk. К ним можно отнести рефакторинг, шаблоны проектирования, карты «класс — обязанности — взаимодействие» и экстремальное программирование в целом.

Как говорилось выше, язык Pascal оказался очень удачным для многочисленных реализаций, в том числе и для объектно-ориентированных вариантов.

#### Delphi

#### Delphi — императивный, структурированный, объектно-ориентированный, высокоуровневый язык программирования со строгой статической типизацией переменных. Основная область использования — написание прикладного программного обеспечения.

Программа Hello World на Delphi:

// Hello World in Delphi

Program Hello\_World;

{$APPTYPE CONSOLE}

Begin

WriteLn('Hello World');

End.

Этот язык программирования является диалектом языка Object Pascal.

При создании языка (и здесь качественное отличие от языка C) не ставилось задачи обеспечить максимальную производительность исполняемого кода или лаконичность исходного кода для экономии оперативной памяти. Изначально язык ставил во главу угла стройность и высокую читаемость, поскольку был предназначен для обучения дисциплине программирования. Эта изначальная стройность в дальнейшем, как по мере роста аппаратных мощностей, так и в результате появления новых парадигм, упростила расширение языка новыми конструкциями.

Так, сложность объектного C++, по сравнению с C, выросла весьма существенно и затруднила его изучение в качестве первого языка программирования, чего нельзя сказать об Object Pascal относительно Pascal.

#### C++

C++ — компилируемый, статически типизированный язык программирования общего назначения.

Программа Hello World на С++:

// Hello World in ISO C++

#include <iostream>

int main()

{

std::cout << "Hello World!" << std::endl;

}

Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком C — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

Язык возник в начале 1980-х годов, когда сотрудник фирмы Bell Labs Бьёрн Страуструп придумал ряд усовершенствований к языку C под собственные нужды. Разрабатывая C с классами, Страуструп написал программу — транслятор, перерабатывающий исходный код C с классами в исходный код простого C. Это позволило работать над новым языком и использовать его на практике. Новый язык, неожиданно для автора, приобрёл большую популярность среди коллег.

Позднее в язык были добавлены новые возможности, такие как виртуальные функции, перегрузка функций и операторов, ссылки, константы, пользовательский контроль над управлением свободной памятью, улучшенная проверка типов. Получившийся язык уже перестал быть просто дополненной версией классического C и был переименован из C с классами в «C++». Его первый коммерческий выпуск состоялся в октябре 1985 года.

C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также компьютерных игр. Существует множество реализаций языка C++, как бесплатных, так и коммерческих и для различных платформ. C++ оказал огромное влияние на другие языки программирования, в первую очередь на Java и C#.

#### Java

Джава — строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle).

Программа Hello World на Java:

// Hello World in Java

class HelloWorld {

static public void main( String args[] ) {

System.out.println( "Hello World!" );

}

}

Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре, для которой существует реализация виртуальной Java-машины. Дата официального выпуска — 23 мая 1995 года. Занимает высокие места в рейтингах популярности языков программирования.

#### C#

Си шарп — объектно-ориентированный язык программирования общего назначения. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core.

Программа Hello World на C#:

//Hello World in C#

class HelloWorld

{

static void Main()

{

System.Console.WriteLine("Hello, World!");

}

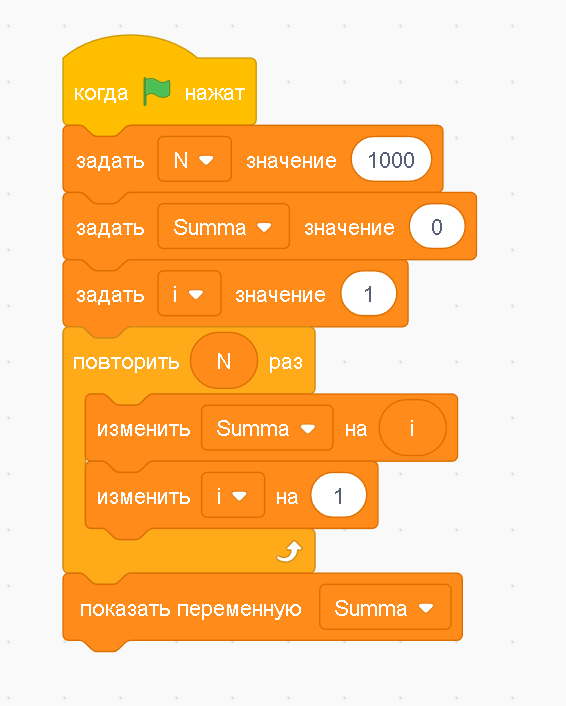
}

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников — языков C++, Delphi, Модула, Smalltalk и, в особенности, Java — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем.

Конечно, языков программирования и было и есть и будет гораздо больше. Упоминать их все, нет ни возможности, ни большого смысла. Однако перейдем к различным классификациям языков программирования.

## Языки программирования: определение, классификации



#include<iostream>

using namespace std;

int main()

{

    int N = 1000;

    int Summa = 0;

    for(int i = 0; i <= N; i++)

        Summa += i;

    cout << Summa;

    return 0;

}

**Язык программирования** (ЯП) — это искусственно созданный язык, основное предназначение которого это запись компьютерных программ. При этом целью применения программы может быть не только компьютер. Это может быть планшет, смартфон, станок с ЧПУ.

**Компьютерная программа** — это набор (последовательность) специальных инструкций, команд составляющих в итоге алгоритм, который в свою очередь должен приводить к конечному результату.

**Команда (инструкция)** — обычно базовое, атомарное указание для вычислительного устройства.

ЯП безусловно делятся на множество самых различных групп и категорий, иногда концепции и принципы построения их отличаются очень заметно. Однако теория языков программирования (есть уже и такой раздел науки на стыке математики и филологии), так вот эта теория выявляет некоторые внутренние составляющие, которые присутствуют у любого языка программирования:

**Алфавит** — очевидно, что под алфавитом ЯП, подразумевается набор символов допустимых в использовании ЯП. На основе алфавита строятся служебные слова ЯП (операторы), а также дополнительные сущности — выражения, команды, идентификаторы. Большинство языков программирования содержат символы латинского алфавита, знаки препинания (. , : ; ! ?), символы математических операций (+ - > < =), а также ряд специальных символов (/ \* $ # @ \_ \ ~ %) и т.д.

**Синтаксис**. Это поверхностные, хотя и вполне строгие правила комбинаций символов алфавита (токенов) в слова, фразы и контекст. Синтаксис языка отвечает за внешнюю «правильность» языка. Синтаксис языка описывает возможные комбинации символов, которые образуют синтаксически правильную программу, а также порядок этих слов.

По аналогии с нашим естественным языком синтаксис определяет и правильность написания слов: «молоко» против «малако», «поезжайте» против «ехайте» и т.п., а также синтаксис определяет правильность порядка слов: «положить яблоко на стол» против «яблоко стол на положить»

**Семантика**. Вводит ряд ограничений на семантические высказывания, в зависимости от контекста. Семантика определяет «внутреннюю» правильность языка. Например, семантика ограничивает использование переменной до ее объявления. Также именно семантика различает два цикла: с предусловием и с постусловием.

По аналогии с естественным языком: иногда набор синтаксически правильных слов приводят к семантически неверному (противоречивому или ложному) результату: «Бесцветные зеленые идеи яростно спят», «Холостой Вася ждал жену домой».

**Словарь** — это набор служебных слов (неотъемлемых от языка). Эти ключевые слова обычно имеют строго определенное значение и назначение. (include, for, switch, auto и т.д.)

### Классификация по машинно-ориентированности

* Машинно-ориентированные (аппаратно-связанные)
* Машинно-неориентированные (аппаратно-несвязанные)

При обзоре истории языков программирования мы упоминали и о машинном языке и о ассемблерах. По большому счету именно они и являются машинно-ориентированными языками, то есть полностью зависящими от архитектуры процессора.

Се остальные языки, в той или иной степени, абстрагируются от конечной реализации аппаратной архитектуры.

### Классификация по трансляции в машинный язык

Все языки кроме машинного (даже ассемблер) являются текстами, написанными на языке относительно понятного человеком, однако совершенно неспособным что-то «объяснить» компьютеру. Для того чтобы этот текст (непосредственно программный код) стал понятным для процессора (для операционной системы), его необходимо «перевести».

Текст программы

Программа Транслятор

Команды на машинном языке

Для этого используется специальная программа — Транслятор. Задача этой программы перевести код программы написанном на любом, кроме машинном, языке как раз в это машинный язык.

Различают два основных типа трансляторов: компиляторы и интерпретаторы.

Впрочем, на сегодняшний день набирают популярность гибридные трансляторы. Когда сначала происходит компиляция в некий промежуточный язык, а потом при непосредственном запуске, происходит интерпретация промежуточного кода в машинный язык «на лету», «в процессе выполнения» — «runtime».

Итак, языки бывают:

* Компилируемые
* Интерпретируемые

Чтобы наглядно объяснить в чем разница компилятора и интерпретатора, представьте такую ситуацию: у нас есть книга на каком-то нам неизвестном языке. Например, на норвежском 😊

У нас есть два пути узнать содержимое этой книги:

Первый, это нанять переводчика, который переведет, перепишет перевод на русском языке и выпустит книгу, которую мы можем взять и читать. Конечно, процесс займет время, но потом мы не привязаны к переводчику. Это пример работы **Компилятора**.

Второй, это постоянно, когда нам захочется почитать эту книгу, приглашать переводчика и он просто будет читать и тут же нам переводить голосом. Книга остается на иностранном языке, но нам постоянно будет необходим переводчик. Это пример **Интерпретатора**.

С одной стороны, кажется компилятор удобнее, однако важно понимать — результат работы компилятора — исполняемый файл (с расширением exe в Windows OS) который состоит из команд на машинном языке, и, если нам потребуется что-то изменить это будет невозможно. Для этого нам потребуется опять компьютер с компилятором и текст исходной программы.

С другой стороны, программу, написанную на интерпретируемом языке, невозможно будет запустить без установленного интерпретатора на этом компьютере, но зато если он установлен и нам что-то понадобится изменить в тексте программы, то мы легко это сделаем.

### Классификация по ориентации на решаемые задачи

Существуют языки, которые предназначены для работы на конкретной платформа, для решения узкого круга задач. Например, языки работающие внутри Систем Управления Базами Данных (СУБД) могут работать только с настройками СУБД и таблицами этих БД. На них написать текстовой редактор или калькулятор не получится. Существуют и универсальные языки, на которых можно выполнить широчайший спектр задач, которые реализованы для различных платформ.