Содержание:

Введение

Мы живем во время невиданных ранее возможностей. Мобильная связь, Глобальная сеть, роботы–пылесосы – о подобных вещах не так давно люди не могли и мечтать. Тем не менее, довольно давно – в начале 19 века появился первый прообраз современных гаджетов – вычислительная машина Беббиджа. Она могла производить крупные математические вычисления – для этого и была создана. Чуть позднее стали появляться и постепенно развиваться первые языки программирования – специальные знаковые системы, обеспечивающие взаимодействие оператора с вычислительной машины.

За прошедшее с 19 века время было изобретено множество языков программирования, различных по назначению, функционалу, сложности освоения и другим параметрам. В целом все они делятся на две большие группы – языки программирования низкого и языки программирования высокого уровня. Первые направлены на прямую коммуникацию с компьютером на близком ему языке, в то время как высокий уровень предполагает большую ориентированность на пользователя или оператора.

Языки программирования высокого уровня очень многообразны. И так как мы живем в «век информационных технологий», их улучшение и создание новых поставлено на широкую ногу. Сегодня каждый программист может выбирать, какой язык ему осваивать и на каком осуществлять непосредственное программирование. Для подробного изучения вопроса выполним цель работы – осуществление обзора языков программирования высокого уровня.

Для достижения данной цели необходимо выполнить следующие задачи:

1. Рассмотреть понятие языка программирования.
2. Проанализировать классификацию языков программирования.
3. Охарактеризовать языки программирования высокого уровня в целом.
4. Осуществить краткий обзор языков программирования высокого уровня.
5. Подвести итоги работы.

Объектом работы выступают языки программирования, а предметом – рассматриваемые языки программирования высокого уровня.

В качестве методологической и теоретической базы работы используются научные работы российских и зарубежных исследователей, в частности таких авторов как Абрамов В. Г., Зуев. Е. А., Моргун А. Н., Фаронов В. В. и других. Авторы, чьи труды используются в работе, заслужили доверие и признание в научных трудах, в частности, в вопросах программирования и изучения языков программирования высокого уровня, так что использование текстов их работ в исследовании целесообразно.

Структура работы включает две главы, каждая из которых, в свою очередь, содержит по два параграфа. Также включены содержание, введение, заключение и список использованных источников.

1. Языки программирования

1.1. Понятие языка программирования

Программу можно представить как набор последовательных команд (алгоритм) для объекта (исполнителя), который должен их выполнить для достижения определенной цели. Так можно «запрограммировать» человека, если составить для него инструкцию «как приготовить оладьи», а он начнет четко ее исполнять. При этом инструкция (программа) для человека будет написана на так называемом естественном языке, например, русском или английском [11].

Обычно принято программировать не людей, а вычислительные машины, используя при этом специальные языки. Использование особых языков вызвано тем, что машины не в состоянии «понимать» наши, т. е. человеческие, языки. Инструкции для машин пишут на языках программирования, которые характеризуются синтаксической однозначностью (например, в них нельзя менять местами определенные слова) и ограниченностью (имеют строго определенный набор слов и символов).

Язык программи́рования – формальная знаковая система, предназначенная для записи компьютерных программ [2]. Язык программирования определяет набор лексических, синтаксических и семантических правил, задающих внешний вид

программы и действия, которые выполнит исполнитель (компьютер) под её управлением.

- Функция: язык программирования предназначен для написания компьютерных программ, которые применяются для передачи компьютеру инструкций по выполнению того или иного вычислительного процесса и организации управления отдельными устройствами.
- Задача: язык программирования отличается от естественных языков тем, что предназначен для передачи команд и данных от человека к компьютеру, в то время как естественные языки используются для общения людей между собой. Можно обобщить определение «языков программирования» – это способ передачи команд, приказов, чёткого руководства к действию; тогда как человеческие языки служат также для обмена информацией.
- Исполнение: язык программирования может использовать специальные конструкции для определения и манипулирования структурами данных и управления процессом вычислений [2].

Первые программы заключались в установке ключевых переключателей на передней панели вычислительного устройства. Очевидно, таким способом можно было составить только небольшие программы.

Первые программы писались на **машинном языке**, т.к. для ЭВМ того времени еще не существовало развитого программного обеспечения, а машинный язык – это единственный способ взаимодействия с аппаратным обеспечением компьютера, так называемым «железом». Каждую команду машинного языка напрямую выполняет то или иное электронное устройство. Данные и команды записывали в цифровом виде (например, в шестнадцатеричной или двоичной системах счисления). Понять программу на таком языке очень сложно; кроме того, даже небольшая программа получалась состоящей из множества строк кода. Ситуация осложнялась еще и тем, что каждая вычислительная машина понимает лишь свой машинный язык [14].

Людям, в отличие от машин, более понятны слова, чем наборы цифр. Стремление человека оперировать словами, а не цифрами привело к появлению **ассемблеров**. Это языки, в которых вместо численного обозначения команд и областей памяти используются словесно–буквенные.

При этом появляется проблема: машина не в состоянии понимать слова. Необходим какой–нибудь переводчик на ее родной машинный язык. Поэтому, начиная со

времен ассемблеров, под каждый язык программирования создаются трансляторы – специальные программы, преобразующие программный код с языка программирования в машинный код. Ассемблеры на сегодняшний день продолжают использовать. В системном программировании с их помощью создаются низкоуровневые интерфейсы операционных систем, компоненты драйверов [9].

После ассемблеров наступил рассвет языков так называемого **высокого уровня**. Для этих языков потребовалось разрабатывать более сложные трансляторы, т. к. языки высокого уровня куда больше удобны для человека, чем для вычислительной машины. В отличие от ассемблеров, которые остаются привязанными к своим типам машин, языки высоко уровня обладают переносимостью. Это значит, что, написав один раз программу на языке программирования высокого уровня, программист может выполнить ее на любом компьютере, если на нем установлен соответствующий ему транслятор [9].

Следующим значимым шагом было появление **объектно–ориентированных языков** программирования, что, в первую очередь, было связано с усложнением разрабатываемых программ. С помощью таких языков программист как бы управляет виртуальными объектами, что в определенном смысле сближает программу с реальностью. На сегодняшний день в большинстве случаев реализация больших и сложных проектов осуществляется с помощью объектно–ориентированных возможностей языков. Хотя существуют и другие современные парадигмы программирования, поддерживаемые другими или теми же языками.

Ранее было сказано, что для перевода кода с языка программирования высокого уровня на машинный язык требуется специальная программа – транслятор. Заложенный в транслятор алгоритм такого перевода сложен. Нам же достаточно знать, что выделяют два основных способа трансляции – **компиляция** программы или ее **интерпретация**.

При компиляции весь исходный программный код (тот, который пишет программист) сразу переводится в машинный. Создается так называемый отдельный исполняемый файл, который никак не связан с исходным кодом. Выполнение исполняемого файла обеспечивается операционной системой (ОС). После того как получен исполняемый файл, для его чтения транслятор уже не нужен.

При интерпретации выполнение кода происходит последовательно (условно можно сказать, строка за строкой). Грубо говоря, операционная система взаимодействует с интерпретатором, а не с файлом, содержащим программный код. Интерпретатор же, прочитав очередной кусок исходного кода, переводит его в машинный (или не совсем машинный, но «понятный» для ОС) и «отдает» его ОС. ОС исполняет этот код и ждет следующей «подачки» от интерпретатора. Питон именно такой язык. Он интерпретируемый язык программирования.

Выполнение откомпилированной программы происходит быстрее, т.к. она представляет собой готовый машинный код. Однако на современных компьютерах снижение скорости выполнения при интерпретации обычно не заметно. Кроме того, интерпретируемые языки обладают рядом преимуществ, среди которых отсутствие подготовительных действий для исполнения программы, что может быть важным для начинающих программировать в первый раз [9].

.

1.2. Классификация языков программирования

Языки программирования можно разделить на два класса: процедурные и непроцедурные. Процедурные (императивные) языки – это языки операторного типа. Описание алгоритма на этом языке имеет вид последовательности операторов. Характерным для процедурного языка является наличие оператора присваивания (Basic, Pascal, C) [3]. Программа, написанная на императивном языке, очень похожа на приказы, выражаемые повелительным наклонением в естественных языках, то есть это последовательность команд, которые должен выполнить компьютер. Программируя в императивном стиле, программист должен объяснить компьютеру, как нужно решать задачу.

Непроцедурные (декларативные) языки – это языки, при использовании которых в программе в явном виде указывается, какими свойствами должен обладать результат, но не говорится, каким способом он должен быть получен. Непроцедурные языки делятся на две группы: функциональные и логические [17].

Декларативные языки программирования – это языки программирования высокого уровня, в которых операторы представляют собой объявления или высказывания в символьной логике. Типичным примером таких языков являются языки логического программирования (языки, основанные на системе правил и фактов). Характерной

особенностью декларативных языков является их декларативная семантика. Основная концепция декларативной семантики заключается в том, что смысл каждого оператора не зависит от того, как этот оператор используется в программе. Декларативная семантика намного проще семантики императивных языков, что может рассматриваться как преимущество декларативных языков над императивными [7].

Рисунок 1 Классификация языков программирования

В программах на языках логического программирования соответствующие действия выполняются только при наличии необходимого разрешающего условия на вывод новых фактов из данных фактов согласно заданным логическим правилам. Логическое программирование основано на математической логике.

Первым языком логического программирования был язык Planner, он был разработан Карлом Хьюитом в Лаборатории искусственного интеллекта Массачусетсского технологического института в 1969 г. В этом языке была заложена возможность автоматического вывода (получения) результата из данных и заданных правил путем перебора вариантов (совокупность которых называлась планом). Но самым известным языком логического программирования является ПРОЛОГ (Prolog), который был создан во Франции в Марсельском университете в 1971 г. Аленом Кольмеро (Colmerauer) [5].

Рисунок 2 Ален Кольмеро

Программа на языке ПРОЛОГ содержит две составные части: факты и правила. Факты представляют собой данные, с которыми оперирует программа, а совокупность фактов составляет базу данных ПРОЛОГа, которая, по сути, является реляционной базой данных. Основная операция, выполняемая над данными, – это операция сопоставления, называемая также операцией унификации или согласования. Правила состоят из заголовка и подцелей. Выполнение программы, написанной на ПРОЛОГе, начинается с запроса и состоит в доказательстве истинности некоторого логического утверждения в рамках заданной совокупности фактов и правил. Алгоритм этого доказательства (алгоритм логического вывода) и определяет принципы исполнения программы, написанной на ПРОЛОГе.

В отличие от программ, составленных на языках процедурного типа, предписывающих последовательность шагов, которые должен выполнять компьютер для решения задачи, на ПРОЛОГе программист описывает факты, правила, отношения между ними, а также запросы по проблеме. Например, пусть у нас есть следующие факты относительно того, кто является чьей мамой:

мама(«Даша»,»Маша»).

мама(«Наташа»,»Даша»).

Кроме этого, имеется правило, вводящее отношение бабушка:

бабушка(X,Y):–

мама(X,Z),

мама(Z,Y) [5].

Теперь мы можем делать запросы на предмет того, кто бабушка того или иного человека, или кто является внучкой (внуком) определенной женщины:

бабушка(«Наташа»,X).

Ответ на этот запрос система ПРОЛОГ выдаст так:

X=Маша [5]

Возможности применения языка ПРОЛОГ весьма обширны. Среди наиболее известных – применение в символической математике, планировании, автоматизированном проектировании, построении компиляторов, базах данных, обработке текстов на естественных языках. Но, наверное, самое характерное применение ПРОЛОГа – это экспертные системы.

На сегодняшний день существует целый класс логических языков; так, от языка Planner также произошли логические языки программирования QA–4, Popler, Conniver и QLISP. Языки программирования Mercury, Visual Prolog, Oz и Fril произошли уже от языка Prolog [5].

Функциональные языки

Первым языком функционального типа является язык ЛИСП, созданный в Массачусетсском технологическом институте в 1956–1959 гг. Джоном Маккарти, который в 1956 г. на Дармутской конференции (США) впервые предложил термин «искусственный интеллект» [12].

Рисунок 3 Джон Маккарти

И хотя до сих пор не утихают споры вокруг этого термина и развившегося научного направления в его рамках, исследователи единодушны в использовании

функциональных и логических языков для данной области. Значительное число работ по искусственному интеллекту реализовано на ЛИСПе.

После своего появления ЛИСПу присваивали много эпитетов, отражающих его черты: язык функций, символьный язык, язык обработки списков, рекурсивный язык. С позиций сегодняшней классификации ЛИСП определяется как язык программирования функционального типа, в основу которого положен метод λ –исчисления [12].

Программа, написанная на функциональном языке, состоит из неупорядоченного набора уравнений, определяющих функции и значения, которые задаются как функции от других значений. Программы и данные ЛИСПа существуют в форме символьных выражений, которые хранятся в виде списковых структур. ЛИСП имеет дело с двумя видами объектов: атомами и списками. Атомы – это символы, используемые для идентификации объектов, которые могут быть числовыми и символьными (понятия, материалы, люди и т.д.). Список – это последовательность из нуля или более элементов, заключенных в круглые скобки, каждый из которых является либо атомом, либо списком. Над списками выполняются три примитивные операции: извлечение первого элемента списка; получение оставшейся части списка после удаления первого элемента; объединение первого элемента списка L и оставшейся части списка Q.

Тексты программ на функциональных языках программирования только описывают способ решения задачи, но не предписывают последовательность действий для решения. В качестве основных свойств функциональных языков программирования обычно рассматриваются следующие: краткость и простота; строгая типизация; модульность; функции – объекты вычисления; чистота (отсутствие побочных эффектов); отложенные (ленивые) вычисления [12].

Кроме ЛИСПа, к функциональным языкам относят РЕФАЛ (разработан в середине 60–х годов В.Ф. Турчиным в МГУ им. М.В. Ломоносова), Haskell, Clean, ML, OCaml, F#.

Приведем пример описания известного алгоритма быстрой сортировки списка на языке Haskell:

```
qsort [] = []

qsort (x:xs) = qsort elts_lt_x ++ [x]

++ qsort elts_greq_x where
```

elts_lt_x = [y | y <– xs, y < x]

elts_greq_x = [y | y <– xs, y >= x]

Здесь записано, что пустой список уже отсортирован. А сортировка непустого списка состоит в том, чтобы разбить список на три: список элементов, меньших головы исходного списка, голова исходного списка ([x]) и список элементов хвоста исходного списка, больше или равных x [4].

Объектно–ориентированные языки

Объектно–ориентированные языки – это языки, в которых понятия процедуры и данных, используемых в обычных системах программирования, заменены понятием «объект» (см. статью «Объектно–ориентированное программирование»). Языком объектно–ориентированного программирования в чистом виде считается SmallTalk, возможности объектно–ориентированного программирования заложены также в Java, C++, Delphi.

Дальнейшее развитие современного программирования связано с так называемым «параллельным программированием». Для реализации этой технологии разрабатываются специализированные объектно–ориентированные языки. К языкам такого типа относят, например, MC# (mcsharp) – высокоуровневый объектно–ориентированный язык программирования для платформы .NET, поддерживающий создание программ, работающих в распределенной среде с асинхронными вызовами [17].

Кроме приведенных видов, известно также самое простое деление – на языки высокого и низкого уровня.

Язык низкого уровня – это язык программирования, предназначенный для определенного типа компьютера и отражающий его внутренний машинный код; языки низкого уровня часто называют машинно–ориентированными языками. Их сложно конвертировать для использования на компьютерах с разными центральными процессорами, а также довольно сложно изучать, поскольку для этого требуется хорошо знать внутренние принципы работы компьютера.

Язык высокого уровня – это язык программирования, предназначенный для удовлетворения требований программиста; он не зависит от внутренних машинных кодов компьютера любого типа. Языки высокого уровня используют для решения

проблем, и поэтому их часто называют проблемно–ориентированными языками. Каждая команда языка высокого уровня эквивалентна нескольким командам в машинных кодах, поэтому программы, написанные на языках высокого уровня, более компактны, чем аналогичные программы в машинных кодах [17].

В первой главе работы рассмотрено понятие и краткая история языков программирования, а также их классификация. Можно сказать, что сегодня существует достаточно много подходов к классификации языков программирования. Тем не менее, деление на языки высокого и низкого уровня является базовым и основополагающим для большинства классификаций.

2. Языки программирования высокого уровня

2.1. Характеристика языков программирования высокого уровня

В информатике , язык программирования высокого уровня является языком программирования с сильной абстракцией из деталей компьютера . В отличии от языков программирования низкого уровня , он может использовать естественный язык элементы , проще в использовании, или может автоматизировать (или даже скрыть полностью) значительные площади вычислительных систем (например , управление памятью), что делает процесс разработки программы проще и более понятно , чем при использовании языка более низкого уровня. Количество абстракции при условии, определяет, как «высокий уровень» язык программирования [17].

В 1960 – х годах, языки программирования высокого уровня , используя компилятор обычно называли autocodes . Примерами являются autocodes COBOL и Fortran .

Первый язык программирования высокого уровня предназначен для компьютеров был планкалкюль , созданный Конрадом Цузе . Тем не менее, он не был реализован в свое время, и его первоначальные взносы были в значительной степени изолированы от других событий из – за Второй мировой войны , в стороне от влияния языка на язык «Superplan» по Хайнц Ратишозер , а также в некоторой степени Алголь . Первый значительно распространенный язык высокого уровня был Фортран, машинно–независимого развитием ранее IBM в AutoCode систем. Алгол,

определенный в 1958 и 1960 год комитетов европейских и американских ученых – компьютерщиков, представил рекурсию , а также вложенные функции в соответствии с лексической областью . Это был также первым язык с четким разграничением между значением и именинами параметрами и соответствующими им семантикой . Алгол также представил несколько структурированных программных концепций, например, в то время– до, и если–то– то в другом месте конструкции и ее синтаксис был первым , который будет описан в формальной записи – « Бэкуса–Наура « (BNF). В течение примерно тот же периода, Кобол введены записи (также называемые структуры) и Лисп введен полностью общая лямбда – абстракция в языке программирования в первый раз [17].

«Язык высокого уровня» относится к более высокому уровню абстракции от машинного языка . Вместо того , чтобы дело с регистрами, адресами памяти и стека вызовов, высокоуровневые языки дело с переменными, массивы, объекты , комплексная арифметика или булевы выражения, подпрограммы и функции, циклы потоков , блокировок и других понятий абстрактной информатики, с акцентом на удобство использования более оптимальной эффективности программы. В отличие от низкоуровневых языков ассемблера , языки высокого уровня имеют мало, если таковые имеются, языковые элементы , которые транслируют непосредственно в родной Компьютера опкодами . Другие функции, такие как обработки строк процедуры, функции языка объектно–ориентированный, и файл ввод / вывод, могут также присутствовать. Одно замечание о языках программирования высокого уровня является то , что эти языки позволяют программисту быть отсоединен и отделен от машины. То есть, в отличие от языков низкого уровня , таких как сборки или на машинном языке, программирования высокого уровня может усилить инструкции программиста и вызывают много движений данных в фоновом режиме , без их ведома. Ответственность и способность выполнять инструкции были переданы машине от программиста [8].

Языки высокого уровня намерены предоставить функции , которые стандартизируют общие задачи, позволяют богатую отладку и поддерживать архитектурный агностицизм; в то время как языки низкого уровня часто производят более эффективный код с помощью оптимизации для конкретной архитектуры системы. Абстракция неустойка является стоимостью , что методы программирования высокого уровня платить за то , что не в состоянии оптимизировать производительность или использовать определенные аппаратные , так как они не воспользоваться некоторыми архитектурными ресурсами низкого уровня. Программирования высокого уровня имеют такие функции, как более

общие структуры данных и операции, интерпретации времени выполнения, а также промежуточные файлы кода; которые часто приводят к исполнению гораздо больше операций , чем это необходимо, более высокое потребление памяти и большего размера двоичной программы. По этой причине, код, который нужно особенно быстро бегать и эффективно может потребоваться использование языка более низкого уровня, даже если язык высокого уровня позволит сделать кодирование проще. Во многих случаях критические части программы в основном на языке высокого уровня могут быть ручными закодировано в языке ассемблера , что приводит к гораздо быстрее, эффективнее, или просто надежно функционировать оптимизированную программу [8].

Однако, с растущей сложностью современных микропроцессорных архитектур, хорошо спроектированные компиляторы для языков высокого уровня часто производят код , сравнимый по эффективности к тому , что большинство программистов низкого уровня могут производить вручную, и тем выше абстракция может позволить для более мощных методов , обеспечивающих более общие результаты , чем их коллеги низкого уровня , в конкретных условиях. Языки высокого уровня разработаны независимо от конкретной архитектуры вычислительной системы. Это облегчает выполнение программы , написанной на таком языке , на любой вычислительной системе с совместимой поддержки интерпретируемого или JIT программы.. В других случаях новые языки высокого уровня развиваются из одного или нескольких других с целью объединения наиболее популярных конструкций с новыми или улучшенными характеристиками. Примером этого является Scala , который поддерживает обратную совместимость с Java , что означает , что программы и библиотеки , написанные на Java , будет по–прежнему использоваться , даже если программирование магазин переходит на Scala; это делает переход проще и срок службы такого высокого уровня кодирования неопределенного. В отличии от программ низкого уровня редко доживают архитектуру системы , которые они были написаны для без серьезного пересмотра. Это техника «Компромисс» для «Абстракции казни».

Достоинства языков программирования высокого уровня:

- Алфавит языка значительно шире машинного, что делает его гораздо более выразительным и существенно повышает наглядность и понятность текста.
- Набор операций, допустимых для использования, не зависит от набора машинных операций, а выбирается из соображений удобства формулирования алгоритмов решения задач определенного класса.

- Конструкции команд (операторов) отражают содержательные виды обработки данных и задаются в удобном для человека виде.
- Используется аппарат переменных и действий с ними.
- Поддерживается широкий набор типов данных [8].

Таким образом, языки программирования высокого уровня являются **машинно–независимыми** и требуют использования соответствующих программ–переводчиков (трансляторов) для представления программы на языке машины, на которой она будет исполняться.

2.2. Краткий обзор языков программирования высокого уровня

Рассмотрим некоторые достойные высокоуровневые языки программирования, которые являются сейчас признанными лидерами и изучение которых способно дать максимальную отдачу.

C#

СИ Шарп является одним из основных языков для написания программного обеспечения под самую популярную операционную систему для настольных компьютеров Windows от Microsoft. Именно в недрах данной компании C# и зародился, используясь в первую очередь для разработки приложения на платформу .NET Framework. С его помощью довольно легко реализовывать взаимодействие с базами данных MS SQL, он прекрасно интегрируется с другими языками (тем самым C и C++) [9].

C# является объектно–ориентированным языком, но поддерживает также и **компонентно–ориентированное** программирование. Разработка современных приложений все больше тяготеет к созданию программных компонентов в форме автономных и самоописательных пакетов, реализующих отдельные функциональные возможности. Главная особенность таких компонентов в том, что они представляют собой модель программирования со свойствами, методами и событиями. У них есть атрибуты, предоставляющие декларативные сведения о компоненте. Они включают в себя собственную документацию. C# предоставляет языковые конструкции, непосредственно поддерживающие такую концепцию работы. Благодаря этому C# подходит для создания и применения программных компонентов.

Вот лишь несколько функций языка C#, обеспечивающих надежность и устойчивость приложений. **Сборка мусора** автоматически освобождает память, занятую недостижимыми неиспользуемыми объектами. **Обработка исключений** предоставляет структурированный и расширяемый подход к обнаружению ошибок и их восстановлению. **Типобезопасная** структура языка делает невозможным чтение из неинициализированных переменных, индексацию массивов за пределами их границ или выполнение непроверенных приведений типов [14].

В C# существует **единая система типов**. Все типы C#, включая типы–примитивы, такие как int и double, наследуют от одного корневого типа object. Таким образом, все типы используют общий набор операций, и значения любого типа можно хранить, передавать и обрабатывать схожим образом. Кроме того, C# поддерживает пользовательские ссылочные типы и типы значений, позволяя как динамически выделять память для объектов, так и хранить упрощенные структуры в стеке.

Чтобы обеспечить совместимость программ и библиотек C# при дальнейшем развитии, при разработке C# много внимания было уделено **управлению версиями**. Многие языки программирования обходят вниманием этот вопрос. В результате программы на этих языках ломаются чаще, чем хотелось бы, при выходе новых версий зависимых библиотек. Вопросы управления версиями существенно повлияли на такие аспекты разработки C#, как раздельные модификаторы virtual и override, правила разрешения перегрузки методов и поддержка явного объявления членов интерфейса.

В недавних версиях C# были использованы другие парадигмы программирования. C# включает функции, поддерживающие приемы функционального программирования, такие как лямбда–выражения. Другие новые возможности поддерживают разделение данных и алгоритмов, например сопоставление шаблонов [14].

C++

Еще один язык высокого уровня из большого семейства C, который в наше время повсеместно используется для решения самых разнообразных задач:

- Создание логических ядер для серьезного ПО;
- Разработки для сетей, серверов и различных служб, которые их обслуживают;
- Для разработки компьютерных игр;
- Для создания интерфейсов программ и многих других задач.

Даже обычный видеоплеер, которых сейчас хоть пруд пруди, чаще всего пишется именно на C++. Да, в нем могут быть элементы интерфейса (кнопки, переключатели и т.п.) разработанные на любом другом языке, но если речь идет о серьезных функциях, например, перекодировка видео в иной формат или прогрузка большого файла порциями, то подобные задачи решает именно C++.

Разработка языка началась в 1979 году. Целью создания C++ было дополнение C возможностями, удобными для масштабной разработки ПО, с сохранением гибкости, скорости и портабельности C. Вместе с тем создатели C++ стремились сохранить совместимость с C: синтаксис первого основан на синтаксисе последнего, и большинство программ на C будут работать и как C++. Изначально новый язык назывался «C с классами», но затем имя было изменено на C++ – это должно было подчеркнуть как его происходжение от C, так и его превосходство над последним [15].

Первый выпуск C++ для коммерческого использования состоялся в 1985 году, вместе с публикацией книги «The C++ Programming Language», которая на долгое время стала его неофициальным стандартом. В 1989 году вышла вторая версия языка в сопровождении книги «The Annotated C++ Reference Manual».

Алфавит C++ включает:

- прописные и строчные латинские буквы и знак подчеркивания;
- арабские цифры от 0 до 9;
- специальные знаки: « { } , | [ ] ( ) + – / % * . \ ‘ : ? < = > ! & # _ ; ^
- пробельные символы: пробел, символы табуляции, символы перехода на новую строку.

Из символов алфавита формируются лексемы (лексема или элементарная конструкция – минимальная единица языка, имеющая самостоятельный смысл):

- ключевые (зарезервированные) слова;
- идентификаторы;
- знаки операций;
- константы;
- разделители (скобки, точка, запятая, пробельные символы);
- комментарии [14].

Java

Замечательный высокоуровневый язык программирования, который активно используется в написании ПО для компьютерной техники. В последние годы в него буквально вдохнула вторую жизнь мобильная ОС Android, программы на которую пишутся в основном именно на Java. Благодаря последнему факту актуальность данного языка ничуть не убывает. Зная его можно писать отличные приложения как для ПК, так и для самых разных современных гаджетов (включая смарт часы и очки виртуальной реальности).

Java разработала компания Sun Microsystems в начале 90–х годов XX века. Ведущую роль в создании языка сыграл канадский инженер Джеймс Гослинг (James Gosling). На ранних этапах разработки язык назывался Oak. Затем его переименовали в честь сорта кофе Java. Связь языка с напитком отражается в логотипе.

Джеймс Гослинг и его единомышленники хотели создать язык с си–подобным синтаксисом. В то же время он должен быть более простым по сравнению с C/C++. Создатели планировали использовать Java для программирования бытовой электроники. Однако практически сразу после выпуска версии 1.0 в 1995 язык стали использовать разработчики серверного и клиентского ПО [13].

В Java реализован механизм управления памятью, который называется сборщиком мусора или garbage collector. Разработчик создаёт объекты, а JRE с помощью сборщика мусора очищает память, когда объекты перестают использоваться. Объясняет эксперт Никита Липский: «Есть такое понятие – циклический мусор. Внутри цикла на все объекты есть ссылки, однако garbage collector в Java удалит его, если объекты не могут использоваться из программы».

Синтаксис языка Java похож на синтаксис других си–подобных языков. Вот его некоторые особенности:

- чувствительность к регистру – идентификаторы User и user в Java представляют собой разные сущности;
- для именования методов используется lowerCamelCase. Если название метода состоит из одного слова, оно должно начинаться со строчной буквы. Пример: firstMethodName();
- для именования классов используется UpperCamelCase. Если название состоит из одного слова, оно должно начинаться с прописной буквы. Пример: FirstClassName.
- название файлов программы должно точно совпадать с названием класса с учётом чувствительности к регистру. Например, если класс называется

FirstClassName, файл должен называться FirstClassName.java;
- идентификаторы всегда начинаются с буквы (A–Z, a–z), знака $ или нижнего подчёркивания _ [15]

По результатам ежегодного отчёта State of the Octoverse, который выпускает Github, язык программирования Java в 2019 году занимает третье место в списке самых популярных.

PHP

Отличный язык, дополняющий современные языки программирования высокого уровня. Он незаменим для разработки самых разнообразных веб–приложений, настройки работы серверного ПО, создания динамичных Интернет–ресурсов, снабженных различными интерактивными функциями, всплывающими окнами и прочими «погремушками». Если Вы видите в своем браузера на любимом сайте появившееся окно с приложением – скорее всего оно написано именно на PHP.

PHP – язык программирования, специально разработанный для написания web–приложений (скриптов, сценариев), исполняющихся на Web–сервере. Синтаксис языка во многом основывается на синтаксисе C, Java и Perl. Он очень похож на C и на Perl, поэтому для профессионального программиста не составит труда его изучить. С другой стороны, язык PHP проще, чем C, и его может освоить веб–мастер, не знающий пока других языков программирования.

Огромным плюсом PHP, в отличие от, например, JavaScript, является то, что PHP–скрипты выполняются на стороне сервера. PHP не зависит от скорости компьютера пользователя или его браузера, он полностью работает на сервере. Пользователь даже может не знать, получает ли он обычный HTML–файл или результат выполнения скрипта [7].

Сценарии на языке PHP могут исполняться на сервере в виде отдельных файлов, а могут интегрироваться в html страницы. PHP способен генерировать и преобразовывать не только HTML документы, но и изображения разных форматов – JPEG, GIF, PNG, файлы PDF и FLASH. PHP способен формировать данные в любом текстовом формате, включая XHTML и XML. PHP – кроссплатформенная технология. Дистрибутив PHP доступен для большинства операционных систем, включая Linux, многие модификации Unix, Microsoft Windows, Mac OS и многих других. PHP поддерживается на большинстве вебсерверов, таких, как Apache, Microsoft Internet Information Server (IIS), Microsoft Personal Web Server и других. Для большинства серверов PHP поставляется в 2–х вариантах – в качестве модуля и в качестве CGI

препроцессора. PHP поддерживает работу с ODBC и большое количество баз данных: MySQL, MSQL, Oracle, PostgreSQL, SQLite и др. Язык программирования PHP, особенно в связке с популярнейшей базой данных MySQL – оптимальный вариант для создания интернет–сайтов различной сложности. Язык PHP постоянно совершенствуется, и ему наверняка обеспечено долгое доминирование в области языков web –программирования [7].

Python

Очень популярный высокого уровня, который применяется для создания сайтов, программного обеспечения. Именно на Python пишется вся логическая система в играх, сложных приложений, налаживается автоматизация самых разнообразных процессов, создаются всевозможные вспомогательные инструменты. Это относительно несложный язык, который часто рекомендуется для начинающих [14].

Python – современный язык программирования, работающий на всех распространенных операционных системах для настольных компьютеров. Язык программирования Питон разрабатывается чуть более 20 лет. В настоящее время активно используется две версии языка – более старая версия 2 и современная версия. Версия 2 более не развивается, но до сих пор еще используется, поскольку очень много программного обеспечения и библиотек разработано именно для версии 2. Между версиями есть существенная несовместимость, в том числе в синтаксисе команд ввода–вывода (программа на языке Python 2–й версии может не работать в 3–й версии и наоборот), но в целом они очень похожи. Мы будет использовать именно версию 3, как более современную и совершенную.

Его достоинства:

1. Кроссплатформенность и бесплатность.
2. Простой синтаксис и богатые возможности позволяют записывать программы очень кратко, но в то же время понятно.
3. По простоте освоения язык сравним с бейсиком, но куда более богат возможностями и значительно более современен.
4. Богатая стандартная библиотека, возможность разработки промышленных приложений (для работы с сетью, GUI, базами данных и т.д.) [14]

Во второй главе работы кратко охарактеризована суть языков программирования высокого уровня, а также осуществлен непосредственно обзор наиболее распространенных сегодня языков такой группы. Рассмотренные языки обладают

множеством достоинств, чаще всего перекрывающих их недостатки, в связи с чем их распространение постепенно увеличивается, как в прикладной программной деятельности, так и в обучении.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была достигнута цель работы – рассмотрены осуществлен обзор языков программирования высокого уровня. Для достижения данной цели были выполнены следующие задачи:

1. Рассмотрено понятие языка программирования.
2. Проанализирована классификация языков программирования.
3. Охарактеризованы языки программирования высокого уровня в целом.
4. Осуществлен краткий обзор языков программирования высокого уровня.

Обзор языков высокого уровня показал, что они действительно разнообразны. В работе были кратко рассмотрены наиболее распространенные сегодня языки программирования высокого уровня, но ими их число далеко не ограничивается. Из современного многообразия возможно осуществить наиболее целесообразный выбор того или иного языка, использование которого будет соответствовать требованиям конкретного проекта.

Время не стоит на месте, и к сегодняшнему дню уже создано немало современных средств программирования – новых языков программирования, специальных сред с использованием шаблонов и т.д. В связи с этим можно сделать вывод, что языки программирования высокого уровня будут продолжать развиваться, расширяя функционал и становясь все более дружественными к оператору и более интуитивно понятными.

В ходе выполнения работы были приобретены новые теоретические знания, которые будут полезны при продолжении обучения, в личной и профессиональной жизни.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Абросимов Л.И. Базисные методы проектирования и анализа сетей ЭВМ. Учебное пособие / Л.И. Абросимов. – М.: Университетская книга, 2015. – 248 с.

2. Абросимова М.А. Информационные технологии в государственном и муниципальном управлении: Учебное пособие / М.А. Абросимова. – М.: КноРус, 2017. – 248 с.
3. Агальцов В.П. Информатика для экономистов: Учебник / В.П. Агальцов, В.М. Титов. – М.: ИД ФОРУМ, НИЦ ИНФРА–М, 2018. –448 с.
4. Байкулов Х.Х. Вопросы проектирования и производства запоминающих устройств / Х.Х. Байкулов, Я.М. Беккер, Б.Д. Платонов. – Л.: ЛДНТП, 2016. – 325 с.
5. Бек Л. Введение в системное программирование / Л. Бек. – М.: Мир, 2016. – 448 с.
6. Боон К. Паскаль для всех / К. Боон. – Москва: Огни, 2015. – 192 с.
7. Гавриков М.М. Теоретические основы разработки и реализации языков программирования: Учебное пособие / М. М. Гавриков, А. Н. Иванченко, Д.В. Гринченков. – М.: КноРус, 2016. – 184 с.
8. Гарет П. Аналоговые устройства для микропроцессоров и мини–ЭВМ / П. Гарет. – М.: Мир, 2015. – 270 с.
9. Гергель В.П. Современные языки и технологии параллельного программирования: Учебник/ предисл.: В.А. Садовничий, В.П. Гергель. – М.: Изд. МГУ, 2016. – 408 с
10. Грызлов В.И. Турбо Паскаль 7.0 / В.И. Грызлов, Т.П. Грызлова. – М.: ДМК, 2016. – 416 с.
11. Гук М. Интерфейсы ПК / М. Гук. – СПб: Питер, 2016. – 416 с.
12. Гук М. Процессоры Intel: от 8086 до Pentium II / М. Гук. – М.: СПб: Питер, 2015. – 224 с.
13. Довек Ж. Введение в теорию языков программирования / Ж. Довек, Ж.–Ж. Леви. – М.: ДМК, 2016. – 134 с.
14. Кетков А. Практика программирования: Бейсик, Си, Паскаль. Самоучитель (+ дискета) / А. Кетков, Ю. Кетков. – М.: БХВ–Петербург, 2016. – 480 с.
15. Мельников С. Delphi и Turbo Pascal на занимательных примерах / С. Мельников. – М.: БХВ–Петербург, 2017. – 419 с.
16. Пелегрен М. Электронные вычислительные машины аналоговые и цифровые / М. Пелегрен. – М.: Машиностроение, 2015. – 408 с.
17. Шевченко В.П. Вычислительные системы, сети и телекоммуникации (для бакалавров) / В.П. Шевченко. – Москва: Огни, 2017. – 980 с