**Теоретическая часть**

**1.ЯП**

**Язы́к программи́рования** — [формальный язык](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA), предназначенный для записи [компьютерных программ](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0). Язык программирования определяет набор [лексических](https://ru.wikipedia.org/wiki/%D0%9B%D0%B5%D0%BA%D1%81%D0%B8%D0%BA%D0%B0), [синтаксических](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) и [семантических](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BC%D0%B0%D0%BD%D1%82%D0%B8%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) правил, определяющих внешний вид программы и действия, которые выполнит исполнитель (обычно — [ЭВМ](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%BE-%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0)) под её управлением. Язык программирования предназначен для написания компьютерных программ, которые представляют собой набор правил, позволяющих компьютеру выполнить тот или иной [вычислительный процесс](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81), организовать управление различными объектами, и т. п. Язык программирования отличается от [естественных языков](https://ru.wikipedia.org/wiki/%D0%95%D1%81%D1%82%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) тем, что предназначен для управления ЭВМ, в то время как естественные языки используются, прежде всего, для общения людей между собой. Большинство языков программирования использует специальные конструкции для определения и манипулирования [структурами данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и управления процессом вычислений. Как правило, язык программирования определяется не только через спецификации *стандарта языка*, формально определяющие его [синтаксис](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) и [семантику](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BC%D0%B0%D0%BD%D1%82%D0%B8%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), но и через *воплощения (реализации) стандарта* — программные средства, обеспечивающих трансляцию или интерпретацию программ на этом языке; такие программные средства различаются по производителю, марке и варианту (версии), времени выпуска, полноте воплощения стандарта, дополнительным возможностям; могут иметь определённые ошибки или особенности воплощения, влияющие на практику использования языка или даже на его стандарт.

Самые востребованные языки программирования:  
  
C  
Java  
Python  
C++  
C#  
JavaScript  
PHP  
R  
Arduino

**2.Критерии выбора ЯП**

**Основными критерия выбора языка программирования являются:**

 **Скорость работы конечного продукта.**  
Требовательным к скорости выполнения могут быть программы с большим объемом математических вычислений, например моделирование физических систем, расчеты большого объема экономических данных, выведение трехмерной графики и прочее. Для данных целей хорошо подойдут компилируемые языки: *ассемблер, С/С++, фортран* и т.д. Почему именно такие? После сборки программа не требует (грубо говоря) ничего лишнего и содержит в себе машинные команды, которые выполняются без лишних задержек. Схема работы таких программ такая: 1) программа исполняется сразу, так сказать она самодостаточна и не требует дополнительных библиотек; 2) программа кроме своего кода содержит вызовы библиотек с машинным кодом (как системных, так и входящих в проект), поэтому, кроме исполнения собственно своих команд, программа вызывает функции из библиотек; 3) в дополнение случаям 1 и 2, программа может работать через прослойку драйверов, которые написаны на языках низкого уровня и работают по умолчанию быстро. Как видно, максимум в схеме возможны 4 блока: *программа -> библиотеки -> драйвера -> железо.*

 **Объем занимаемой оперативной памяти.**  
Данное требование появляется, когда программа разрабатывается для встраиваемых систем, мобильных платформ, микроконтроллеров и так далее. В данных случаях, чем меньше памяти расходует программа на данном языке – тем лучше. К таким языкам, опять же, относятся *ассемблер, С/С++, Objective-C* и другие. Список языков подобен списку пункта 1, так как чем меньше функциональных блоков в схеме исполнения, тем меньше занимается и памяти компьютера.  
Если данное требование некритично, то можно использовать «истинно высокоуровневые языки».

 **Скорость разработки программы.**  
Данное требование возникает тогда, когда начальник говорит «программа нужна не позже, чем вчера!» или еще какая срочность. Тогда выбор падает на высокоуровневые языки с максимально человеколюбивым синтаксисом. Это, например, *Java, Flash* и подобные. На данных языках время разработки может существенно сокращаться из-за обилия сторонних библиотек, максимально «очеловеченного» синтаксиса, и подобных вещей. Скорость выполнения программ, написанных на данных языках страдает, причем порой весьма ощутимо. Схема выполнения на примере Java:  
*Программа в виде байт-кода -> виртуальная машина-анализатор -> системные библиотеки -> драйвера -> железо.*  
Самым медленно работающим блоком в данной схеме является анализатор – он должен байт-код программы транслировать «на лету» в машинный код, при этом потратив много времени на точное определение инструкции. Поэтому быстрая разработка зачастую означает медленное выполнение.

 **Ориентированность на компьютер или человека**  
С кем будет работать программа в первую очередь? С человеком, или с компьютером? В пером случае программа должна обладать мощной графической частью, отвечающей требованиям дизайна и юзабилити. Разработка графической части зачастую требует достаточно много времени, т.к. отличается немалой сложностью. Здесь сложность возникает в том, что вывод графики – это немало математики, а значит присутствует требовательность к скорости исполнения, а из-за сложности разработки присутствует необходимость в высокоуровневом языке. В данном случае, на мой взгляд, очень хорошо подходит *С++/C#* с их одновременной и высокоуровневостью, и скоростью выполнения программ на них. Однако, если ГИП не очень сложный, но красивый, возможно использование *Java* и *Flash*, на которых создание красивых интерфейс гораздо проще, нежели на С++ и, тем более, С. Если программа ориентирована в первую очередь на «скрытую работу» с минимумом взаимодействия с пользователем, тогда выбор должен ложиться в сторону быстрых языков (*ASM, C*)

 **Кроссплатформенность.**  
Кроссплатформенность – возможность работы программы на различных платформах, в различных ОС с минимальными изменениями. В этой сфере можно выделить такие языки: *Java, C#,Flash,C++* с различными библиотеками и другие, менее используемые, языки.  
Java создавался с тем условием, что программы на данном языке должны работать на любой платформе, где есть *JVM* – *Java Virtual Machine*. Программы на Java вообще не требуют никаких изменений – после компиляции получается .jar файл, который будет работать и на Windows, и на Mac OS, и на Linux и еще немало где. Аналогичная ситуация и с Flash, только список платформ гораздо менее обширный. С С++ дело обстоит труднее. На чистом С++ написать кроссплатформенную программу довольно трудно, у кода возникает обширная избыточность, теряется достоинство в скорости выполнения. Облегчают задачу кроссплатформенные библиотеки, например, [Qt](http://qt.nokia.com/), которые позволяют добиться принципа «один код на все платформы», однако на каждую платформу нужно программу собирать отдельно (при этом разными компиляторами).  
В этот раздел так же можно включить интерпретируемые, скриптовые языки – для их работы нужно наличие интерпретатора языка в системе. Данные языки очень удобны в плане разработки, но достаточно медлительны. Схема их работы напоминает схему работы Java/Flash, только анализатор стал еще медленнее – полумашинный байт код анализировать «на лету» гораздо проще, чем человеческий код. Так же, это влечет к большему потреблению памяти.

 **Скорость внесения изменений, скорость тестирования**  
Проект стремительно развивается, в него постоянно вносятся изменения, порой немало? Тогда выбор должен падать на высокоуровневые языки, где любой функциональный блок можно быстро переписать. Для подтверждения – я думаю, гораздо проще дебажить тот же С++, чем ассемблер. А еще проще Java. Но тут очень много тонкостей, которые таятся даже не сколько в языке, сколько в разработчике с его стилем программирования и компиляторах. Тем не менее, язык вносит свою долю в это дело, так или иначе упрощая/осложняя работу программиста.

**3.Стили программирования**

**Паради́гма программи́рования** — это совокупность идей и понятий, определяющих стиль написания [компьютерных программ](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0) (подход к программированию). Это способ [концептуализации](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%86%D0%B5%D0%BF%D1%82%D1%83%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_(%D0%BB%D0%B8%D0%BD%D0%B3%D0%B2%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0)), определяющий организацию вычислений и структурирование работы, выполняемой компьютером.

Парадигма программирования не определяется однозначно языком программирования; практически все современные языки программирования в той или иной мере допускают использование различных парадигм ([мультипарадигмальное программирование](https://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BF%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5" \o "Мультипарадигмальное программирование)). Так, на [языке Си](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), который не является объектно-ориентированным, можно работать в соответствии с принципами объектно-ориентированного программирования, хотя это и сопряжено с определёнными сложностями; функциональное программирование можно применять при работе на любом императивном языке, в котором имеются функции, и т. д.

Также существующие парадигмы зачастую пересекаются друг с другом в деталях (например, модульное и объектно-ориентированное программирование), поэтому можно встретить ситуации, когда разные авторы употребляют названия из разных парадигм, говоря при этом, по сути, об одном и том же явлении.

**Основные модели программирования**

* [Императивное программирование](https://ru.wikipedia.org/wiki/%D0%98%D0%BC%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
* [Декларативное программирование](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BA%D0%BB%D0%B0%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
* [Структурное программирование](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
* [Функциональное программирование](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
* [Логическое программирование](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
* [Объектно-ориентированное программирование](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
  + [Компонентно-ориентированное программирование](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
  + [Прототипно-ориентированное программирование](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D1%82%D0%B8%D0%BF%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
  + [Агентно-ориентированное программирование](https://ru.wikipedia.org/wiki/%D0%90%D0%B3%D0%B5%D0%BD%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)

**4. Оформление листинга**

**Станда́рт оформле́ния ко́да** (**станда́рт коди́рования**, **стиль программи́рования**) — набор правил и соглашений, используемых при написании [исходного кода](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) на некотором [языке программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). Наличие общего стиля программирования облегчает понимание и поддержание исходного кода, написанного более чем одним программистом, а также упрощает взаимодействие нескольких человек при разработке программного обеспечения.

Стандарт оформления кода обычно принимается и используется некоторой группой разработчиков [программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) для единообразного оформления совместно используемого кода. Целью принятия и использования стандарта является упрощение восприятия программного кода человеком, минимизация нагрузки на память и зрение при чтении программы.

Образцом для стандарта кодирования может стать набор соглашений, принятых в какой-либо распространённой печатной работе по языку (например, стандарт кодирования на языке [Си](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), получивший сокращённое наименование K&R, происходит из [классического описания](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%A1%D0%B8_(%D0%BA%D0%BD%D0%B8%D0%B3%D0%B0)) Си его авторами — [Керниганом](https://ru.wikipedia.org/wiki/%D0%9A%D0%B5%D1%80%D0%BD%D0%B8%D0%B3%D0%B0%D0%BD,_%D0%91%D1%80%D0%B0%D0%B9%D0%B0%D0%BD" \o "Керниган, Брайан) и [Ритчи](https://ru.wikipedia.org/wiki/%D0%A0%D0%B8%D1%82%D1%87%D0%B8,_%D0%94%D0%B5%D0%BD%D0%BD%D0%B8%D1%81" \o "Ритчи, Деннис)), широко применяемая библиотека или API (так, на распространение [венгерской нотации](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%BD%D0%B3%D0%B5%D1%80%D1%81%D0%BA%D0%B0%D1%8F_%D0%BD%D0%BE%D1%82%D0%B0%D1%86%D0%B8%D1%8F) явно повлияло её использование в [MS-DOS](https://ru.wikipedia.org/wiki/MS-DOS) и [Windows API](https://ru.wikipedia.org/wiki/Windows_API" \o "Windows API), а большинство стандартов кодирования для [Delphi](https://ru.wikipedia.org/wiki/Delphi_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)" \o "Delphi (язык программирования)) используют, в той или иной мере, манеру кодирования библиотеки [VCL](https://ru.wikipedia.org/wiki/Visual_Component_Library)).

Реже разработчик языка выпускает подробные рекомендации по кодированию. Например, выпущены стандарты кодирования на [C#](https://ru.wikipedia.org/wiki/C_Sharp) от [Microsoft](https://ru.wikipedia.org/wiki/Microsoft" \o "Microsoft) и на [Java](https://ru.wikipedia.org/wiki/Java" \o "Java) от [Sun](https://ru.wikipedia.org/wiki/Sun_Microsystems" \o "Sun Microsystems). Предложенная разработчиком или принятая в общеизвестных источниках манера кодирования в большей или меньшей степени дополняется и уточняется в корпоративных стандартах.