|  |
| --- |
| **Содержание**  **Введение** **1**  **1.1** Основные определения ПО 2  **1.2** Виды классификации ПО 3  **1.3** Система программирования 4  **1.4** Этапы подготовки программы 4  **1.5** Разница между языками 4  **Заключение** **5**  **Список литературы 6**  Приложение 7 |

**Основные определения ПО**

**Програ́ммное обеспе́чение** (**ПО**) — программа или множество программ, используемых для управления компьютером.

Имеются и другие определения :

1. совокупность программ системы обработки информации и программных документов, необходимых для эксплуатации этих программ
2. компьютерные программы, процедуры и, возможно, соответствующая документация и данные, относящиеся к функционированию компьютерной системы.

Программное обеспечение является одним из видов обеспечения вычислительной системы, наряду с техническим (аппаратным), математическим,

информационным, организационным, методическим и правовым обеспечением.

**Виды классификации ПО**

**Программное обеспечение делится на 3 категории:**

**1)**прикладное;  
**2)**системное;  
**3)**инструментальное;

**Системное** — программа, предназначенная для эксплуатации и технического обслуживания ЭВМ, управления и организации вычислительного процесса при решении любой конкретной

**Прикладное** —  [программа](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), ориентированная на решение конкретных задач,  рассчитанная на взаимодействие с пользователем. В большинстве [ОС](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) прикладные программы не могут обращаться к ресурсам [компьютера](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80)  напрямую, взаимодействуя с оборудованием и другими программами через ОС.

**Инструментарное –** программа, используемые как инструмент при создании документов в повседневной деятельности.

**Система програмирования**

**Система программирования** — это система для разработки новых программ на конкретном языке программирования.

Современные системы программирования обычно предоставляют пользователям  **мощные и удобные средства разработки программ.**  В них входят:

1. **Компилятор;**
2. **Интегрированная среда для разработки;**
3. **Библиотека стандартны программ;**
4. **Многооконный режим работы;**
5. **Встроенный ассемблер;**
6. **Встроенная справочная служба;**

**Этапы подготовки программы**

Этап 1 – Определение проблемы

Этап 2 – Выработка требований

Этап 3 – Создание плана разработки

Этап 4 – Разработка архитектуры системы

Этап 5 – Детальное проектирование

Этап 6 – Кодирование и отладка

Этап 7 – Тестирование компонентов

Этап 8 – Интеграция компонентов

Этап 9 – Тестирование всей системы

Этап10 – Сопровождение, внесение изменений и оптимизация.

**Этап 1 – Определение проблемы**

Перед тем как приступать к кодированию, необходимо четко сформулировать проблему, которую ваша будущая программа должна решать.

На данном этапе проводится простая формулировка сути проблемы без каких-либо намеков на ее возможные решения, при этом формулировать ее следует на языке, понятном пользователю, т.е. она должна быть описана с пользовательской точки зрения.

**Этап 2 – Выработка требований**

**Требования к программе** – это подробное описание всех возможностей программы и действий, которые должна выполнять программа.

Требования вырабатывают для того, чтобы свести к минимуму изменения системы после начала непосредственной разработки. Такие требования должны быть обязательно официальными, т.е. документально оформлены. Так как это гарантирует то, что функциональность системы определяется заказчиком, а не программистом.

Выработка требований очень важна, так как она позволяет определить функциональность программы до начала программирования.

**Этап 3 – Создание плана разработки**

На данном этапе Вы уже должны в формальном виде составить план разработки программного обеспечения с учётом существующей проблемы и выработанных требований. Иными словами, Вы должны составить план того, как Вы будете действовать дальше.

**Этап 4 – Разработка архитектуры системы или высокоуровневое проектирование**

Данный этап также очень важный, так как, не имея хорошей архитектуры, Вы можете решать правильную проблему, но прийти к неправильному решению. Хорошая архитектура программы упрощает программирование, а плохая архитектура усложняет его.

Архитектура системы обычно включает:

* Общее описание системы;
* Основные компоненты;
* Формат и способ хранения данных;
* Специфические бизнес-правила;
* Способ организации пользовательского интерфейса;
* Подход к безопасности системы;
* Оценки производительности;
* Возможности масштабирования;
* Моменты, связанные с интернациональностью, т.е. будет ли система интернациональной.

**Этап 5 – Детальное проектирование**

На этом этапе проводится проектирование программы на низком уровне, иными словами, здесь проектируются классы и методы, рассматриваются, оцениваются и сравниваются различные варианты и причины выбора окончательных подходов и способов реализации.

При разработке небольших программ программисты обычно сами проектируют программу на таком уровне, это выглядит как написание псевдокода или рисование схем, поэтому часто этот этап рассматривается как часть непосредственного кодирования и в таких случаях итоговый документ (если того требует формальность) состоит преимущественно из различных набросков и заметок программистов.

Но при реализации крупных проектов данному процессу отводится отдельный этап и проектирование в этом случае проводится с очень высокой степенью детальности.

**Этап 6 – Кодирование и отладка**

После того как код написан, программисту необходимо отладить этот код, чтобы в нем не было никаких ошибок.

**Этап 7 – Тестирование компонентов**

После того, как код написан, и проведена отладка, необходимо провести тестирование реализованного функционала. Если программа состоит из нескольких компонентов, сначала тестируют каждый компонент в отдельности, так как очень крупные программы включают огромный функционал, который часто разделяют на отдельные компоненты, разработка которых осуществляется по отдельности. В менее крупных проектах этот этап может включать просто тестирование отдельных классов.

**Этап 8 – Интеграция компонентов**

Когда тестирование всех компонентов закончено, можно переходить к интеграции всех компонентов в единый программный комплекс, этот этап как раз и подразумевает процесс интеграции, т.е. слияния всех компонентов в единую систему.

В небольших проектах этот этап может заключаться в объединении нескольких классов, на что будет затрачено не больше одного дня, но в крупных проектах этот этап может длиться не один месяц.

**Этап 9 – Тестирование всей системы**

На данном этапе проводится тестирование всей системы, уже с учётом интеграции всех компонентов. На этом этапе можно выявить проблемы взаимодействия компонентов и устранить их. Также на этом этапе основным предметом тестирования является безопасность, производительность, утечка ресурсов и другие моменты, которые невозможно протестировать на более низких уровнях тестирования.

**Этап 10 – Сопровождение, внесение изменений и оптимизация**

После запуска программы в промышленную эксплуатацию осуществляется сопровождение этой программы, т.е. внесение изменений на основе выявленных недочетов в процессе эксплуатации системы, а также проводится оптимизация функционала или добавление нового.

**Разница между языками**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **языки** | **функциональность** | **Логическое** | **процедурность** | **оринтерование** |
| **1** | **python** |  | **-** | **+** | **+** |
| **2** | **ada** | **-** | **-** | **+** | **+** |
| **3** | **C++** |  | **-** | **+** | **+** |
| **4** | **C#** |  | **-** | **+** | **+** |
| **5** | **C** | **-** | **-** | **+** | **+** |
| **6** | **Java** | **-** | **-** | **+** | **+** |
| **7** | **Haskell** | **+** |  | **+** | **-** |
| **8** | **Common**  **LISP** | **+** |  | **+** | **+** |
| **9** | **Smalltalk** | **+** |  | **+** | **+** |
| **10** | **Delphi** |  | **-** | **+** | **+** |
| **11** | **Prolog** |  | **+** | **-** | **-** |

**Приложения:**

1. <http://book.kbsu.ru/theory/chapter6/1_6.html>
2. [Ссылка](https://unix-it.ru/information/klassifikatsiya-programmnogo-obespecheniya-kompyutera/#:~:text=%D0%92%20%D0%B7%D0%B0%D0%B2%D0%B8%D1%81%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D0%B8%20%D0%BE%D1%82%20%D0%BD%D0%B0%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D1%8F%2C%20%D0%B0,%D0%B8%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%80%D0%B8%D0%B9%20%D0%B4%D0%BB%D1%8F%20%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.)
3. [Система программирование](http://book.kbsu.ru/theory/chapter6/1_6_11.html)
4. [Этапы подготовки программы при разработке](https://info-comp.ru/programmirovanie/724-stages-of-program-development.html)