Классификация программного обеспечения.

Совокупность программ и сопровождающая их документация, предназначенная для решения задач на ПК называется **программным обеспечением (ПО)**.

Системное ПО – необходимо для управления ПК, для создания и поддержки выполнения других программ пользователя, а также для предоставления пользователю набора всевозможных услуг.

Операционная система (ОС) – совокупность программ, управляющих работой всех устройств ПК и процессом выполнения прикладных программ.

Сервисные системы расширяют возможность операционных систем, предоставляя набор дополнительных услуг пользователю.

Оболочка ОС – программный продукт, который делает общение пользователя с ПК более комфортным.

Утилиты – служебные программы, предоставляющие пользователю дополнительные услуги. К ним относятся: дисковые компрессоры, дисковые дефрагментаторы, программы резервного копирования данных, архиваторы, антивирусы.

Различие между операционными оболочками и операционными средами достаточно условные. В ряде литературных источников между ними различия нет, т.к. операционная среда обладает всеми признаками оболочки, за исключением того, что последнее не формирует новой среды выполнения программ – это является функцией лишь ОС. В свою очередь операционную среду нельзя назвать ОС, т.к. она не может функционировать

самостоятельно, следовательно, операционная среда – полнофункциональная надстройка над ОС.

Программно-инструментальные средства – программные продукты, предназначенные для разработки ПО. К ним относятся системы программирования, которые включают систему команд процессора, периферийных устройств, трансляторы (компиляторы и интерпретаторы) с различных языков программирования.

В настоящее время наиболее часто используют процедурно-ориентированные системы программирования. Кроме того, используют системы программирования, не требующих описания алгоритма обработки данных, такие как SQL, использующуюся в Access и других программах. При их использовании указываются исходные данные и требуемые результаты, а сам алгоритм генерируется системой программирования.

Система технического обслуживания – совокупность программно-аппаратных средств для обнаружения сбоев в процессе работы ПК, а также отдельных блоков, узлов и является инструментом специалистов по эксплуатации и ремонту технических средств ПК.

Прикладное ПО – предназначено для расширения определённых классов задач пользователя. Отличительной чертой проблемно-ориентированных программных продуктов является их сравнительно узкая направленность на определённый круг решаемых задач и большое их разнообразие.

Пакеты общего назначения предназначены для решения типовых задач обработки данных.

Интегрированные ПП – совокупность функционально-различных программных модулей, способных взаимодействовать между собой путём обмена данными через единый пользовательский интерфейс.

В структуре пакета предусмотрен модуль управления, обеспечения переключения между приложениями и бесконфликтное использование общих данных. Современные интегрированные пакеты содержат как правило 5 функциональных компонентов: СУБД, графический редактор, табличный процессор, текстовый редактор, коммуникационные средства.

Общая характеристика технологии программных средств.

Решение задачи па ЭВМ – процесс получения результатов на основе обработки исходных данных с помощью программы, составленной из команд.

Под программно-инструментальными средствами будем понимать компоненты ПО, позволяющие программировать решение задач. К программно-инструментальным средствам относятся:

- Языки программирования и соответствующие им трансляторы,
- СУБД с языковыми средствами программирования в их среде;
- Электронные таблицы с соответствующими средствами их настройки.

Появление принципиально новых по сравнению с алгоритмическими языками программно-инструментальных средств изменило традиционное представление о процессе программирования и программе. Таким образом, под **программным средством** будем понимать программу или иное формализованное описание, обеспечивающее автоматизацию решения задач на ЭВМ.

Принципиальная схема разработки программных средств. (Технология, процесс создания).

1. Постановка задачи
2. Математическое описание
3. Разработка алгоритма
4. Составление программы
5. Тестирование и отладка
6. Приёмо-сдаточные испытания
7. Опытная эксплуатация
8. Промышленная эксплуатация

На I этапе раскрывается сущность задачи, т.е. формулируется цель её решения, определяется взаимосвязь с другими задачами, устанавливаются состав и формы представления входной, промежуточной и результативной информации. Характеризуются формы, методы контроля достоверности информации на ключевых этапах решения задачи. Специфицируются формы взаимодействия пользователя и ЭВМ.

На II этапе технического процесса разработки программ выполняется формализованное описание задачи. Т.е. устанавливаются и формируются средствами языка математики логико-математические зависимости между исходными данными и результатом.

При выборе метода решения задачи предпочтение отдаётся методу, который наиболее полно удовлетворяет следующим требованиям:

1. Обеспечивает необходимую точность полученных результатов и не обладает свойством вырождения (т.е. бесконечного зацикливания) на каком-либо участке задачи при определённом наборе исходных данных.
2. Позволяет использовать готовые стандартные программы.
3. Способствует наиболее быстрому получению искомых результатов.

III этап представляет собой алгоритмизацию решения задачи, т.е. разработку оригинального или адаптацию (уточнение или корректировка) уже известного алгоритма.

Алгоритмизация - сложный творческий процесс, в его основу положено фундаментальное понятие математики и программирования – **алгоритм**.

Алгоритм – точное предписание, определяющее вычислительный процесс, ведущий от варьирующих начальных данных к искомому результату.

Любой алгоритм обладает следующими свойствами:

1. **Детерминированность**(определённость, однозначность) – означает, что набор указаний алгоритма должен быть однозначно и точно понят любым исполнителем. Это свойство определяет однозначность результата работы алгоритма при одних и тех же исходных данных.
2. **Массовость** – предполагает возможность варьирования исходных данных в определённых пределах. Это свойство определяет пригодность использования алгоритма для решения множества задач данного класса.
3. **Результативность** – означает, что для любых допустимых исходных данных он должен через конечное число шагов (итераций) завершить работу.
4. **Дискретность** – возможность разбиения определенного алгоритмического процесса на отдельные элементарные действия, возможность реализации которых человеком или ЭВМ не вызывает сомнения, а результат вполне определён и понятен.

Сложность и ответственность этого этапа для решения одной и той же задачи как правило существование множества различных алгоритмов, отличающихся друг от друга уровнями сложности, объёмами вычислительных и логических операций, составом необходимой исходной и промежуточной информации, точность полученных результатов и другими факторами, которые могут оказать существенное влияние на эффект выбранного способа решения задачи.

Схема процесса алгоритмизации решения задачи.

1. Выделение автономных этапов процесса решения задачи (как правило с одним входом и выходом).
2. Формализованное описание содержания работ, выполняемых на каждом выделенном этапе.
3. Проверка правильности реализации выбранного алгоритма на различных примерах решения задачи.

Способы описания алгоритмов.

1. **Словесный** – отражает содержание выполнимых действий средствами естественного языка. **Достоинства:** общая доступность, возможность описывать алгоритм с любой степенью детализации. **Недостаток:** громоздкое описание, отсутствие строгой формализации в силу неоднозначности восприятия естественного языка.
2. **Формульно-словесный** – основан на записи содержания выполняемых действий с использованием изобразительных возможностей языка математики, дополненного необходимыми пояснениями средствами естественного языка.
3. **Графический** – представляет собой изображение логико-математической структуры алгоритма, при которой все этапы процесса обработки информации отображаются с помощью набора геометрических фигур, имеющих строго определённую конфигурацию, в соответствии с предписанным им характером выполняемых действий. Изображение схем алгоритмов осуществляется по определённым правилам, ГОСТам, ОСТам.

Описание алгоритма средствами языка операторных схем – это изображение последовательности операций процесса обработки данных с помощью заданного набора буквенных символов, обозначающих ту или иную типовую операцию.

Последовательность выполнения операций алгоритма определяется расположением операторов в схеме. Из-за малой наглядности и информативности отображение процесса решения задачи использование языка операторных схем не нашло широко практического применения.

Описание алгоритма с помощью таблиц решения.

Таблица решения, описывая задачу и необходимые для её решения действия в наглядной форме, определяет какие условия должны быть выполнены прежде, чем можно будет переходить к тому или иному действию.

Лёгкость освоения специалистом любой области, простота модификации, комплектность и, главное, более общая по сравнению со схемами алгоритма, форма представления информации – основные достоинства таблиц решения.

IV этап представляет собой составление (адаптация, кодирование) программ.

Процесс кодирования заключается в переводе описания алгоритма на один из доступных ЭВМ языков программирования.

V этап – тестирование и отладка. Оба этих процесса функционально связаны между собой, хотя их цели несколько отличаются друг от друга.

Тестирование – совокупность действий, предназначенных для демонстрации правильности работы программы в заданных диапазонах изменения внешних условий и режимов эксплуатации программы.

Цель тестирования заключается в демонстрации отсутствия (или выявления) ошибок в разработанных программах на заранее подготовленном наборе контрольных примеров.

Процессу тестирования сопутствует понятие **отладка**, которое подразумевает совокупность действий, направленных на устранение ошибок в программах, начиная с момента обнаружения фактов ошибочной работы программы и завершая устранением причин их возникновения.

По своему характеру ошибки в программах делятся на синтаксические и логические.

Синтаксическая ошибка – некорректная запись отдельных языковых конструкций с точки зрения правил их представления для выбранного языка программирования. Эти ошибки выявляются автоматически, при трансляции исходной программы до её выполнения. После устранения синтаксических ошибок проверяется логика работы программ на исходных данных. При этом возможны следующие проявления (основные формы) логических ошибок:

а) в какой-то момент программа не может продолжить работу (возникает программное прерывание, обычно сопровождающееся указанием места в программе, где оно произошло).

б) программа работает, но не выдаёт всех запланированных результатов, и не выходит на остановку (происходит её зацикливание).

в) программа выдаёт результаты и завершает свою работу, но они полностью или частично не совпадают с контрольными.

Программа считается отлаженной, если она безошибочно выполняется на достаточно представительном наборе тестовых данных, обеспечивающих проверку всех её участков (ветвей).

Процесс тестирования и отладки носит итерационный характер и считается одним из наиболее трудоёмким.

По оценке специалистов он может составлять от 30% до 50% в общей структуре затрат времени на разработку проектов, и зависит от объёма и логической сложности программ.

Для сокращения затрат на проведение тестирования и отладки широко применяются специальные программные средства тестирования (генераторы тестовых данных) и приёмы отладки (метод троссировки программ, позволяющий выявить, все ли ветви программы были задействованы при решении задачи с заданным набором исходных данных).

После завершения тестирования и отладки программные средства вместе с сопровождающей документацией передаются пользователю для эксплуатации.

Состав документации обычно оговаривается заказчиком и разработчиком на этапе подготовки технического задания на программные средства. Обычно эти документы регламентируют работу пользователя в процессе эксплуатации программы, а также содержащие информацию о программе, необходимые в случае возникновения потребности внесения изменений и дополнений в неё.

VI и VII этапы. При передаче пользователю программных средств создаётся специальная комиссия, включающая в свой состав представителей заказчиков и разработчиков.

Комиссия, в соответствии с заранее составленном и утверждённым планом проводит работы по приёмке-передаче программных средств и сопроводительной документации.

По завершении работы комиссия оформляет акт приёмки-передачи.

При реализации сложных программ по согласованию пользователя разработчиком этап эксплуатации разбивается на 2 подэтапа:

1. Экспериментальная (опытная);
2. Промышленная.

Смысл экспериментальной эксплуатации заключается во внедрении разработанных программ на объекте заказчика с целью проверки их работоспособности и удобства работы пользователей при решение реальных задач (обычно не менее года). После завершения этого периода и устранения выявленных при этом ошибок программа передаётся в промышленную эксплуатацию.

Для повышения качества работ разработчик может по договорённости с пользователем осуществить сопровождение программ.

Описанная схема технологического процесса разработки ПО отображает их жизненный цикл, т.е. временной интервал с момента зарождения программы до момента отказа от её эксплуатации.

Технология системного проектирования программных средств.

Принципиальная схема разработки.

1. Описание требований к системе и её отдельным компонентам.
2. Математическое описание.
3. Детальная спецификация на программные модули.
4. Логическое проектирование компонентов системы.
5. Кодирование и автономная отладка программы.
6. Комплексная отладка системы.
7. Приемо-сдаточные испытания. Опытная эксплуатация, промышленная эксплуатация.

Системный аналитик, исходя из общих целей назначения, технических характеристик, состава и описания требований пользователя к системе формирует общие формальные требования ПО системы.

Специалист системо-техник преобразует общие формальные требования в детальные спецификации на отдельные программы. Участвует в логическом проектирование компонентов системы, определяет общую структуру.

Прикладной программист преобразует спецификацию в логическую структуру программных модулей, а затем в программный код.

Системный программист обеспечивает сопряжение программных модулей с программной средой, в рамках которой предстоит функционировать программным средствам.