Средства создания программных продуктов

**Содержание**

Введение

Раздел I. Разработка программ для ЭВМ

I.1 Этапы планирования программы

I.2 Понятие и особенности алгоритмов

Раздел II. Средства, используемые для создания программ

II.1 Язык программирования – инструмент для разработки программы. Виды и классификация языков программирования

II.2 Структурное и объектно-ориентированное программирование

Заключение

Список использованной литературы

**Введение**

Современный этап развития общества характеризуется возрастающей ролью информационной сферы, представляющей собой совокупность информации, информационной инфраструктуры, субъектов, осуществляющих сбор, формирование, распространение и использование информации, а также системы регулирования возникающих при этом общественных отношений. Информационная сфера, являясь системообразующим фактором жизни общества, активно влияет на состояние политической, экономической, оборонной и других составляющих безопасности Российской Федерации.

В современном обществе компьютер играет огромную роль, уже трудно представить труд ученых, инженеров, экономистов, бухгалтеров без использования вычислительной техники. Но компьютер сам по себе не способен даже на простые операции, поэтому для того чтобы человек мог использовать компьютер необходимо так называемое программное обеспечение. Программа руководит ресурсами компьютера и предоставляет их в распоряжение пользователя.

Программы, работающие на компьютере, можно разделить на три категории:

* прикладные программы, непосредственно обеспечивающие выполнение необходимых пользователям работ: редактирование текстов, рисование картинок, обработка информационных массивов и т. д.;
* системные программы, выполняющие различные вспомогательные функции, например создание копии используемой информации, выдачу справочной информации о компьютере, проверку работоспособности устройств компьютера и т. д.;
* вспомогательное ПО (инструментальные системы и утилиты)

Понятно, что грани между указанными тремя классами программ весьма условны, например, в состав программы системного характера может входить редактор текстов, т. е. программа прикладного характера.

Для работы на компьютере разработаны и используются сотни тысяч различных прикладных программ для различных применений. Наиболее широко применяются программы:

* подготовки текстов (документов) на компьютере – редакторы текстов;
* подготовки документов типографского качества – издательские системы;
* обработки табличных данных – табличные процессоры;
* обработки массивов информации – системы управления базами данных..

Прикладная программа – это любая конкретная программа, способствующая решению какой-либо задачи в пределах данной проблемной области.

Например, там, где на компьютер возложена задача контроля за финансовой деятельностью какой-либо фирмы, прикладной будет программа подготовки платежных ведомостей.

Прикладные программы могут носить и общий характер, например, обеспечивать составление и печатание документов и т.п.

Прикладные программы могут использоваться либо автономно, то есть решать поставленную задачу без помощи других программ, либо в составе программных комплексов или пакетов.

Следует отметить, что программное обеспечение, в то числе и прикладное разрабатывается с помощью специальных инструментов – языков программирования специалистами в этой области.

**Раздел I. Разработка программ для ЭВМ**

**I.1 Этапы планирования программы**

Решение любой задачи на ЭВМ представляет собой процесс обработки данных с помощью программы. Создание такой программы предполагает выполнение ряда последовательных этапов:

* постановка задачи;
* математическое описание и выбор метода;
* разработка алгоритма решения;
* составление программы;
* тестирование и отладка программы;
* эксплуатирование программы.

Первый этап представляет собой постановку задачи. На этом этапе формулируется цель задачи, определяется взаимосвязь с другими задачами, раскрывается состав и форма представления входной, промежуточной и результативной информации, характеризуются формы и методы контроля достоверности информации на ключевых этапах решения задачи, определяются формы взаимодействия пользователя с ЭВМ в ходе решения задачи и т.п.

На втором этапе разработки программы выполняется формализованное описание программы, т.е. устанавливаются и формулируются средства языка математики логико-математические зависимости между исходными и результатными данными. Для задач, допускающих возможность математического описания, необходимо выбрать численный метод решения, а для нечисловых задач – принципиальную схему решения в виде однозначно понимаемой последовательности выполнения элементарных математических и логических операций.

Третий этап подготовки решения задачи представляет собой алгоритмизацию ее решения, т.е. разработку оригинального или адаптацию известного алгоритма. Алгоритмизация – это сложный процесс, носящий в значительной степени творческий характер. Постановка задачи и ее алгоритмизация составляют до 20-30% общего времени на разработку программы. Сложность и ответственность реализации данного этапа объясняется тем, что для решения одной и той же задачи, как правило, существует множество различных алгоритмов.

Алгоритм – это точное предписание, определяющее вычислительный процесс, ведущий от варьируемых начальных данных к искомому результату. Это конечный набор правил, однозначно раскрывающих содержание и последовательность выполнения операций для систематического решения определенного класса задач за конечное число шагов.

Четвертый этап – составление программы. На этом этапе производится перевод описания алгоритма на один из доступных для ЭВМ языков описания.

Тестирование и отладка составляют заключительный этап разработки программы решения задачи на ЭВМ. Оба эти процесса функционально связаны между собой, хотя их цели несколько отличаются друг от друга.

Тестирование представляет собой совокупность действий, предназначенных для демонстрации правильной работы программы. Цель тестирования заключается в выявлении возможных ошибок в разработанных программах путем их проверки на наборе заранее подготовленных контрольных примеров.

Процессу тестирования сопутствует процесс отладки, который подразумевает совокупность действий, направленных на устранение ошибок в программе. Действия по отладке начинаются с момента обнаружения фактов ошибочной работы программы и завершаются устранением причин, порождающих ошибки.

После завершения процессов тестирования и отладки программные средства вместе с сопроводительной документацией передаются пользователю для эксплуатации. Основное назначение сопроводительной документации – обеспечить пользователя необходимыми инструктивными материалами по работе с программой.

**I.2 Понятие и особенности алгоритмов**

«Алгоритм» - одно из фундаментальный понятий информатики, а также математики. Происхождение самого термина связано с искаженным представлением (Algorithmi) имени средневекового арабского математика Мухаммеда аль-Хорезми (787 – 850). В то время алгоритмами называли описанные в трактатах аль-Хорезми правила арифметики (сложение, вычитание, умножение столбиком и деление уголком многозначных чисел) и десятичная система счисления.

В настоящее время понятие алгоритма трактуется шире. Это понятие применимо ко всем областям человеческой деятельности.

Алгоритмы встречаются не только в вычислительной технике, но и в обыденной жизни. Примеры алгоритмов из обыденной жизни:

* поездка в институт;
* ремонт телевизора (по инструкции);
* поиск пропавшей вещи;
* выращивание растений на участке и т.п.

Не все задачи могут быть решены с помощью алгоритмов. Например, написание музыки, написание стихов, научное открытие. Компьютер используется для решения лишь тех задач, для которых может быть составлен алгоритм.

Любой алгоритм обладает следующими свойствами:

* детерминированность,
* массовость,
* результативность,
* дискретность.

Детерминированность (определенность) означает, что набор указаний алгоритма должен быть однозначно понят любым исполнителем. Это свойство определяет однозначность результата работы алгоритма при заданных исходных данных.

Массовость алгоритма предполагает возможность варьирования исходных данных в некоторых пределах. Это свойство определяет пригодность использования алгоритма для решения множества конкретных задач определенного класса.

Результативность алгоритма означает, что для любых допустимых исходных данных он должен через конечное число шагов (или итераций) завершить свою работу.

Дискретность алгоритма означает возможность разбиения определенного алгоритмического процесса на отдельные элементарные этапы, возможность реализации которых человеком или компьютером не вызывает сомнения, а результат выполнения каждого элементарного этапа вполне определен и понятен.

Таким образом, алгоритм дает возможность чисто механически решать любую конкретную задачу из некоторого класса однотипных задач. Существует несколько способов описания алгоритмов:

* словесный,
* формально-словесный,
* графический и др.

Словесный способ описания алгоритма отражает содержание выполняемых действий средствами естественного языка. К достоинствам этого способа описания следует отнести его общедоступность, а также возможность описывать алгоритм с любой степенью детализации. К главным недостаткам этого способа следует отнести достаточно громоздкое описание, отсутствие строгой формализации вследствие неоднозначности восприятия естественного языка.

Формально-словесный способ описания алгоритма основан на записи содержания выполняемых действий с использованием изобразительных возможностей языка математики, дополненного с целью указания необходимых пояснений средствами естественного языка. Данный способ, обладая всеми достоинствами словесного способа, вместе с тем более лаконичен, а значит, и более нагляден, имеет большую формализацию, однако тоже не является строго формальным.

Графический способ описания алгоритмов представляет собой изображение логико-математической структуры алгоритма, при котором все этапы процесса обработки данных представляются с помощью определенного набора геометрических фигур (блоков), имеющих строго определенную конфигурацию в соответствии с характером выполняемых действий. (см. рис.1)

* начало, конец
* вычисления
* ввод / вывод
* проверка условия
* модификация
* подпрограмма

Рис. 1. Основные графические обозначения блоков программ.

Все блоки в схеме располагаются в последовательности сверху вниз и слева направо, объединяясь между собой линиями потока.

Приведем пример: математическая постановка задачи - задано квадратное уравнение:

,

где , b, c – некоторые параметры. Вычислить его действительные корни.

Решение:

;

Если D≥0, то уравнение имеет 2 действительных корня

.

Если D<0, то уравнение не имеет действительных корней.

Блок-схема алгоритма решения:

**Раздел II. Средства, используемые для создания программ**

Для создания программ или программных продуктов используются аппаратное и программное обеспечение.

Под аппаратным обеспечением понимают обычно все узлы, модули и блоки, составляющие компьютер или компьютерную систему. В современных компьютерах используется так называемая «открытая архитектура», т.е. состав аппаратного обеспечения компьютера можно изменить, поменяв один из модулей, или расширить, вставив дополнительный модуль.

Аппаратное обеспечение современных ПК включает в себя следующее:

* системный блок,
* устройства ввода информации в ПК (например, клавиатура),
* устройства вывода информации из ПК (например, монитор).

Системный блок, клавиатура и монитор вместе составляют персональный компьютер в минимальной конфигурации, т.е. позволяют работать с информацией на компьютере.

О программном обеспечении, позволяющем создавать программные продукты (фактически те же программы), пойдет речь ниже.

**II.1 Язык программирования – инструмент для разработки программы. Виды и классификация языков программирования**

Языком программирования называют способ записи алгоритмов (решений различных задач) на языке, понятном для ЭВМ.

Процессор компьютера может обрабатывать информацию, представленную лишь в виде машинных кодов (двоичных). Запись команд непосредственно на языке, понятном процессору, довольно утомительна. Такие программы создавались программистами лишь для первых ламповых ЭВМ.

В 1950-х годах появились первые языки, называемые Автокодами, а позднее они стали называться Ассемблеры. Здесь переменные величины стали обозначаться символами, а команды зарезервированными наборами символов. Таким образом, процесс программирования стал более понятен для человека. Такое представление работы с данными было реализовано с помощью специального переводчика – транслятора – программы, переводящей текст создаваемой человеком программы в машинные команды.

Языки типа Ассемблеров (языки низкого уровня) являются машинно-ориентированными, т. е. для каждого типа процессора существует свой язык-Ассемблер.

Создание языка программирования заключается в создании программы-транслятора этого языка в машинные коды.

Различают 2 типа трансляции:

1. компиляция заключается в полном предварительном переводе всего набранного пользователем текста программы в программу машинных кодов, причем последняя сохраняется в памяти, откуда и происходит ее выполнение; такая программа-компилятор загружается в память только перед исполнением компиляции;

2. интерпретация заключается в последовательном чтении транслятором очередной команды, переводе ее в машинный код и последующем ее выполнении, при этом результаты предыдущих переводов в памяти не сохраняются; программа-интерпретатор постоянно находится в оперативной памяти компьютера.

Следует отметить, что откомпилированная пользовательская программа выполняется быстрее, чем интерпретированная. Поэтому такие языки программирования высокого уровня, как Turbo Pascal, Фортран, Си, используют компиляцию при реализации программ.

Каждый язык программирования характеризуется элементами:

1) Алфавит – набор символов, разрешенных к использованию и воспринимаемых компилятором. С помощью этих символов строятся команды, переменные, операции.

2) Синтаксис – правила записи команд и операций.

3) Семантика – смысловое содержание и способы построения конструкций языка.

Алгоритмические языки представляют собой средства описания данных и алгоритмов решения задач, они разработаны для составления программы пользователем. В настоящее время разработано большое количество языков программирования. Они отличаются друг от друга различными свойствами и областью применения.

Класс машинно-зависимых языков представлен ассемблером. Язык ассемблера делает доступными все программно-управляемые компоненты компьютера, поэтому он применяется для написания программ, использующих специфику конкретной аппаратуры. Ассемблер – это наиболее трудоемкий язык программирования, и из-за его низкого уровня не удается построить средства отладки, которые существенно снизили бы трудоемкость разработки программ. Программирование на ассемблере требует от программиста детальных знаний технических компонент персонального компьютера. Ассемблер используется в основном для системного программирования.

К классу машинно-ориентированных языков можно отнести языки группы С, С++, Турбо С. Эти языки являются результатом попытки объединить возможности ассемблера со встроенными структурами данных.

Класс универсальных языков программирования представлен наиболее широко: Бейсик, Фортран, Паскаль и др.

Исторически одним из самых распространенных языков стал Бейсик. Он прост в освоении и использовании. Написать на этом языке программу в 20-30 строк и получить результат можно за несколько минут. Для различных типов ПК разработаны различные версии языка Бейсик.

Паскаль является одним из самых распространенных, хотя он и создавался как учебный. Использование в структуре языка специального кода позволило в 4-5 раз уменьшить длину текста программы и в 4-5 раз увеличить быстродействие программы. Версия Паскаля для ПК – Турбо-Паскаль – характеризуется такими важными особенностями, как полноэкранное редактирование и управление, графика, звуковое сопровождение и развитые связи с DOS. Система программирования на Турбо-Паскале является резидентной программой. Это позволяет пользователю вводить тексты программ и немедленно их выполнять, не тратя времени на компилирование.

Язык Кобол был разработан специально для решения экономических задач. Он дает возможность составлять наиболее удобочитаемые программы, которые понятны и непрограммисту. В обработке данных сложной структуры Кобол бывает эффективнее Паскаля.

Фирмой IBM в развитие идей Фортрана, Алгола и Кобола был предложен язык PL/1, который получил наибольшее распространение на больших машинах. PL/1 разрабатывался как универсальный язык программирования, поэтому он располагает большим набором средств обработки цифровой и текстовой информации. Однако эти достоинства делают его весьма сложным для обучения и использования.

Класс проблемно-ориентированных языков программирования представлен языками Лого, РПГ и системой программирования GPSS. Язык Лого был создан с целью обучения школьников основам алгоритмического мышления и программирования. Лого – диалоговый процедурный язык, реализованный на основе интерпретатора с возможностью работы со списками и на их основе с текстами, оснащенными развитыми графическими средствами, которые доступны для детского восприятия. Этот язык реализован в большинстве ПК, применяемых в школах.

РПГ, или генератор отчетов, представляет собой язык, включающий многие понятия и выражения, которые связаны с машинными методами составления отчетов и проектирования форм выходных документов. Язык используется главным образом для печати отчетов, записанных в одном или нескольких файлах баз данных.

Система программирования GPSS ориентирована на моделирование систем с помощью событий. В терминах этого языка легко описывается и исследуется класс моделей массового обслуживания и другие системы, работающие в реальном масштабе времени.

В последние годы развивается объектно-ориентированный подход к программированию. Наиболее полно он реализован в языках Форт и СМОЛТОК. Форт сочетает в себе свойства операционной системы, интерпретатора и компилятора одновременно. Основной чертой языка является его открытость. Программист может легко добавлять новые операции, типы данных и определения основного языка. Форт позволяет поддерживать многозадачный режим работы, использует принцип одновременного доступа программ. К объектно-ориентированным средам разработки программ можно отнести Delphi, Visual Basic, Visual FoxPro.

К функциональнымязыкам программирования можно отнести языки Лиеп, Пролог И Снобол. Лиеп является инструментальным средством для построения программ с использованием методов искусственного интеллекта. Особенность этого языка заключается в удобстве динамического создания новых объектов. В качестве объектов могут выступать и сами исходные объекты. В настоящее время для Лиепа определились две сферы активного применения: проектирование систем искусственного интеллекта и анализ текстов на естественном языке.

Нетрудно заметить, что языка, который был бы идеальным для всех случаев, не существует. Какой язык является лучшим, надо определять в каждой конкретной ситуации. Поэтому перед разработкой программы следует установить:

1. назначение разрабатываемой программы;
2. время выполнения программы;
3. ожидаемый размер программы – хватит ли объема памяти?
4. необходимость сопряжения программ с другими пакетами или программами;
5. возможность и необходимость переноса программы на другие типы компьютеров;
6. основные типы данных, с которыми будет работать программа;
7. характер и уровень использования в программе аппаратных средств (дисплея, клавиатуры, НМД и др.);
8. возможность и целесообразность использования стандартных библиотек программ.

**II.2 Структурное и объектно-ориентированное программирование**

Рассмотрим идею структурного и объектно-ориентированного программирования с использованием языка Pascal.

Возможность структурного программирования в ТП 7.0 заключается в разделении программы на меньшие программы, оформив последние как процедуры и функции.

При создании процедур или функций пытаются достичь целей:

1. разделения некоторой большой общей задачи на несколько меньших по объему и сложности частных задач;

2. уменьшения объема программы за счет многократного применения команд, формируемых в виде отдельных процедур или функций;

3. универсализации применения процедур или функций, то есть решение частной задачи обобщается с той целью, чтобы ее можно было использовать для решения широкого круга задач.

Таким образом, структурное программирование привело к разделению труда среди программистов. Это значит, что каждый программист может заниматься разработкой конкретной частной задачи, поставленной перед ним. Ему не обязательно знать настоящих размеров проекта, так как его задача может составлять лишь малую часть последнего.

Структура процедуры, функции:

|  |
| --- |
| Раздел объявлений и соглашений |
| PROCEDURE (FUNCTION) | Имя процедуры (функции) со списком параметров; |
| LABEL | Подраздел объявления локальных меток; |
| CONST | Подраздел объявления локальных констант; |
| TYPE | Подраздел объявления локальных типов; |
| VAR | Подраздел объявления локальных переменных; |
| Раздел вложенных процедур и функций |
| PROCEDURE (FUNCTION) | Имя вложенной процедуры (функции) |
| LABEL | Подраздел объявления локальных меток; |
| CONST | Подраздел объявления локальных констант; |
| TYPE | Подраздел объявления локальных типов; |
| VAR | Подраздел объявления локальных переменных; |
| BEGIN |  |
| Раздел основного блока процедуры или функции |
| END; |  |

Непосредственно текст процедуры располагается в разделе объявлений и соглашений основной программы перед оператором BEGIN основной программы. Завершается описание процедуры оператором END;. Начинается блок описания процедуры (функции) оператором BEGIN.

Процедура может содержать также при необходимости и вложенные процедуры, описанные согласно структуре.

Все переменные, метки, константы, типы, описанные внутри процедуры (функции), называемые локальными, доступны только внутри ее, но не в основной программе. Глобальные переменные, типы, константы, метки можно использовать внутри процедуры (функции).

Объявление процедуры:

PROCEDURE <Имя процедуры> (Список параметров);

Список параметров может отсутствовать.

Объявление функции:

FUNCTION <имя функции>(Список параметров):<Тип возвращаемого результата>;

Список параметров может отсутствовать.

Вызов процедуры (функции) в основной программе выполняется командой:

<Имя процедуры>(Список параметров);

<Имя функции>(Список параметров);

Если в описании процедуры (функции) список параметров отсутствует, то вызов последней выполняется просто командой:

<Имя процедуры>;

<Имя функции>;

Пример: PROCEDURE INPUT(VAR P1:Integer; P2,P3:Byte);

Здесь: P1 – параметр-переменная ,P2,P3 – параметры-значения, INPUT – имя процедуры.

Перед параметром-переменной всегда ставится VAR. Это значит, что при выполнении процедуры параметр P1 может изменяться в отличие от параметров P2,P3, поглощаемых процедурой.

Рассмотрим пример задачи: сложение двух целых чисел.

1 вариант решения задачи

Program Summa;

Uses Crt;

Var A,B,S: LongInt;

Procedure SSS(Var SUM:LongInt; A,B:LongInt);

Begin

SUM:=A+B;

End;

Begin {Основной блок программы}

ClrScr;

Write(‘Введите числа а и b=’);

ReadLn(A,B);

{Вызов процедуры SSS для поиска суммы}

SSS(S,A,B); {В глобальную переменную S считывается значение локальной переменной Sum вызываемой процедуры, значения А и В «поглощаются» процедурой для суммирования}

WriteLn(‘Сумма А+В=’,S);

ReadKey;

End.

2 вариант решения задачи

Program Summa;

Uses Crt;

Var A,B,S: LongInt;

Function SSS(A,B:LongInt):LongInt;

Begin

SSS:=A+B;

End;

Begin {Основной блок программы}

ClrScr;

Write(‘Введите числа а и b=’);

ReadLn(A,B);

{Вызов функции SSS для поиска суммы}

S:=SSS(A,B); {В переменную S считывается значение функции SSS, переменные A,B – параметры, необходимые для суммирования}

WriteLn(‘Сумма А+В=’,S);

ReadKey;

End.

Идея современного подхода к программированию – объектно-ориентированного – заключается в объединении данных и обрабатываемых их процедур в единое целое - объект. Таким образом, программа представляется в виде совокупности объектов, каждый из которых есть реализация некоторого класса (типа).

Весь окружающий мир состоит из объектов (предметов живой и неживой природы), а объекты, используемые в программировании, являются высшим уровнем абстракции данных.

Приведем пример: пусть задан некоторый класс (тип) под названием «геометрический объект», включающий в себя подклассы геометрических фигур «плоская» и «объемная». В свою очередь, подклассы могут подразделяться на подклассы: «с вершинами», «без вершин». Подкласс «без вершин» может подразделятся на «окружность» и «эллипс». Таким образом, представлена иерархия объектов, причем здесь соблюдается следующее правило: свойства и методы, определенные ранее в более общем объекте (в зависимости от уровня), наследуются последующими (более специфическими объектами).

В основе ООП лежат три основных понятия:

- наследование: данный объект может наследовать свойства объекта и добавлять свойства, характерные только для данного объекта; в рассмотренном выше примере объект «геометрическая фигура» является объектом-предком, включающий в себя остальную иерархию объектов-потомков. Здесь все свойства и методы объекта-предка могут быть использованы объектами-потомками.

- инкапсуляция: объединение данных и методов (процедур и функций) в одно целое – объект;

- полиморфизм: при описании наследников некоторого объекта существует возможность модифицировать методы объекта-предка внутри объекта-наследника, т. е. объект-наследник может иметь методы с такими же именами как и объект-предок, но выполняющие другие функции.

Объект – определяемый пользователем тип, содержащий в себе как данные, так и подпрограммы для их обработки. Данные объекта называют полями, а подпрограммы для их обработки – методами объекта.

Инкапсуляцией называют объединение данных и методов обработки в одном типе.

Данные типа объект в программе описываются в разделе описания типов TYPE:

TYPE

{Описание полей объекта}

<Имя>=OBJECT

<поле 1>:<Тип 1>;

<поле 2>:<Тип 2>;

………………..

<поле N>:<Тип N>;

{Описание методов объекта}

PROCEDURE <Имя метода 1>[(параметры)];

……………………..

FUNCTION <Имя метода N>[(параметры)]:<Тип>;

END;

Требования, предъявляемые к полям и методам объекта:

1) Необязательное: поля объекта обрабатываются только с помощью его методов.

2) Обязательное: поля объекта являются параметрами по умолчанию для его методов, т.е. не допускается указание в списке параметров-переменных с такими же именами как и поля.

Описание методов объектов производится по составному имени:

имя объекта. имя метода:

<имя объекта>.<имя метода> [(параметры)];

Begin

<список команд>;

End;

Переменная типа объект называется экземпляром объекта.

Переменная типа объект описывается в разделе описания переменных VAR:

VAR <имя экземпляра>:<имя объекта>;

В основной программе доступ к полям и методам экземпляра объекта производится по составному имени:

<имя экземпляра>:<имя метода>; - обращение к методу экземпляра,

<имя экземпляра>:<имя поля>; - обращение к полю (переменной) экземпляра.

**Заключение**

Даже при наличии десятков тысяч программ для ЭВМ пользователям может потребоваться что-то такое, чего не делают (или делают, но не так) имеющиеся программы. В этих случаях следует использовать системы программирования, т.е. системы для разработки новых программ.

Современные системы программирования для персональных компьютеров обычно предоставляют пользователю весьма мощные и удобные средства для разработки программ. В них входят:

* компилятор, осуществляющий преобразование программ на языке программирования в программу машинных кодах, или интерпретатор, осуществляющий непосредственное выполнение текста программы на языке программирования высокого уровня;
* библиотеки программ, содержащие заранее подготовленные программы, которыми могут пользоваться программисты;
* различные вспомогательные программы, например отладчики, программы для получения перекрестных ссылок и т.д.

Системы программирования, прежде всего, различаются, естественно, по тому, какой язык программирования они реализуют. Среди программистов пишущих программы для персональных компьютеров, наибольшей популярностью пользуются языки Си, Си++, Паскаль, Visual Basic, Delphi.

**Список использованной литературы**

1. Информатика и математика для юристов: Учебное пособие для вузов (Гриф МО РФ) / Под ред.проф. Х.А.Андриашина, проф. С.Я.Казанцева – М.:Юнити-Дана, 2006.- 463 с.
2. Информатика и математика: Учебник / Под ред. Д.В. Захарова, 2007.
3. Информатика: Учебник (Гриф МО РФ) / Каймин В.А.,2-е изд. перераб. И доп.- М: Инфра-М., 2004.- 272 с.
4. Информатика: Учебник (Гриф МО РФ) / Под ред. Н.В.Макаровой, 3-е изд., перераб. и доп. – М.: Финансы и статистика, 2007.- 768 с.
5. Информатика: Учебник для вузов (Гриф МО РФ) / Острейковский В.А., М: Высшая школа, 2001.- 511 с.
6. Математика и информатика / Турецкий В.Я. – 3-е изд., испр. И доп. – М.: Инфра-М, 2004.- 560 с.
7. Основы информатики и математики для юристов. В 2-х томах: краткий курс в таблицах, схемах и примерах. Учебник / Богатов Д.Ф., Богатов Ф.Г.- М.: Приор, 2004.- 144 с., 166 с.
8. Бобровский С.И. Delphi. Учебный курс – СПб.: Питер, 2005. – 736 с.: ил.
9. Турбо Паскаль 7.0 – К.: Торгово-издательское бюро BHV, 2006 – 448 с.: ил.
10. Семакин И.Г., Шестаков А.П. Основы программирования: Учебник. – М.: Мастерство; НМЦ СПО; Высшая школа, 2003. – 432 с.
11. Дагене В.А., Григас Г.К., Аугутис К.Ф. 100 задач по программированию. – М.: Просвещение, 2003.
12. Гофман В., Хомоненко А Delphi. – СПб.: БХВ – Санкт-Петербург, 2006. – 800.