**МЕТОДИЧЕСКА РАЗРАБОТКА ПО ИНФОРМАТИКЕ**

***Системы программирования. Основные понятия и определения***

**Оглавление**

Введение 3

1. Система программирования как неотъемлемая часть современных

ЭВМ 4

1.1 Понятие системы программирования и её функции 4

1.2 Языки программирования 6

2. Классификация и компоненты систем программирования 8

2.1 Классификация систем программирования 8

2.1.1 Машинно-ориентированные системы программирования 8

2.1.2 Машинно-независимые системы программирования 11

2.2 Средства создания программ 13

Заключение 15

Список использованных источников 16

**Введение**

В пятидесятые годы двадцатого века с появлением компьютеров на электронных лампах началось бурное развитие систем программирования.

К сегодняшнему дню насчитывают несколько поколений систем программирования. Каждое из последующих поколений по своей функциональной мощности качественно отличается от предыдущего. С появлением персональных компьютеров системы стали составными частями интегрированных сред разработки. Появились системы, применяемые в различных офисных программах.

В настоящее время системы программирования применяются в самых различных областях человеческой деятельности, таких как научные вычисления, системное программирование, обработка информации, искусственный интеллект, издательская деятельность, удаленная обработка информации, описание документов.

С течением времени одни системы развивались, приобретали новые черты и остались востребованы, другие утратили свою актуальность и сегодня представляют в лучшем случае чисто теоретический интерес.

На данном этапе важным остается то, что необходимо владеть информацией о системе программирования, как части современных ЭВМ а также знать терминологию и уметь ею пользоваться.

**Цель** работы - изучить систему программирования, как неотъемлемую часть современных ЭВМ.

**Задачи:**1. Рассмотреть понятие и функции системы программирования;

2. Описать существующие языки программирования;

3. Изучить виды систем программирования и средства создания программ;

**1. Системы программирования как неотъемлемая часть ЭВМ**

**1.1 Понятие системы программирования и её функции**

Неотъемлемая часть современных ЭВМ – системы программного обеспечения, являющиеся логическим продолжением логических средств ЭВМ, расширяющим возможности аппаратуры и сферу их использования. Система программного обеспечения, являясь посредником между человеком и техническими устройствами машины, автоматизирует выполнение тех или иных функций в зависимости от профиля специалистов и режимов их взаимодействия с ЭВМ. Основное назначение программного обеспечения – повышение эффективности труда пользователя, а также увеличение пропускной способности ЭВМ посредством сокращения времени и затрат на подготовку и выполнение программ [4].

Программное обеспечение ЭВМ можно подразделить на общее и специальное программное обеспечение.

Общее программное обеспечение реализует функции, связанные с работой ЭВМ, и включает в себя системы программирования, операционные системы, комплекс программ технического обслуживания.

Специальное программное обеспечение включает в себя пакеты прикладных программ, которые проблемно ориентированы на решение вполне определенного класса задач [1].

Системой программирования называется комплекс программ, предназначенный для автоматизации программирования задач на ЭВМ.

Система программирования освобождает проблемного пользователя или прикладного программиста от необходимости написания программ решения своих задач на неудобном для него языке машинных команд, и предоставляют им возможность использовать специальные языки более высокого уровня. Для каждого из таких языков, называемых входными или исходными, система программирования имеет программу, осуществляющую автоматический перевод (трансляцию) текстов программы с входного языка на язык машины [2].

Важно различать язык программирования и реализацию языка. Язык – это набор правил, определяющих систему записей, составляющих программу, синтаксис и семантику используемых грамматических конструкций. Реализация языка – это системная программа, которая переводит (преобразует) записи на языке высокого уровня в последовательность машинных команд [5].

Имеется два основных вида средств реализации языка: компиляторы и интерпретаторы.

Компилятор транслирует весь текст программы, написанной на языке высокого уровня, в ходе непрерывного процесса. При этом создается полная программа в машинных кодах, которую затем ЭВМ выполняет без участия компилятора.

Интерпретатор последовательно анализирует по одному оператору программы, превращая при этом каждую синтаксическую конструкцию, записанную на языке высокого уровня, в машинные коды и выполняя их одна за другой. Интерпретатор должен постоянно присутствовать в зоне основной памяти вместе с интерпретируемой программой, что требует значительных объемов памяти [1].

Следует заметить, что любой язык программирования может быть как интерпретируемым, так и компилируемым, но в большинстве случаев у каждого языка есть свой предпочтительный способ реализации. Языки Фортран, Паскаль в основном компилируют; язык Ассемблер почти всегда интерпретирует; языки Бейсик и Лисп широко используют оба способа.

Основным преимуществом компиляции является скорость выполнения готовой программы. Интерпретируемая программа неизбежно выполняется медленнее, чем компилируемая, поскольку интерпретатор должен строить соответствующую последовательность команд в момент, когда инструкция предписывает выполнение. В то же время интерпретируемый язык часто более удобен для программиста, особенно начинающего. Он позволяет проконтролировать результат каждой операции. Особенно хорошо такой язык подходит для диалогового стиля разработки программ, когда отдельные части программы можно написать, проверить и выполнить в ходе создания программы, не отключая интерпретатора [1].

Для построения языков программирования используется совокупность общепринятых символов и правил, позволяющих описывать алгоритмы решаемых задач и истолковывать смысл созданного написания.

**1.2 Языки программирования**

Ядро системы программирования составляет язык. Существующие языки программирования можно разделить на две группы: процедурные и непроцедурные.

Процедурные (или алгоритмические) программы представляют из себя систему предписаний для решения конкретной задачи. Роль компьютера сводится к механическому выполнению этих предписаний.

Процедурные языки разделяют на языки низкого и высокого уровня.

Языки низкого уровня (машинно-ориентированные) позволяют создавать программы из машинных кодов, обычно в шестнадцатиричной форме. С ними трудно работать, но созданные с их помощью высококвалифицированным программистом программы занимают меньше места в памяти и работают быстрее. С помощью этих языков удобнее разрабатывать системные программы, драйверы (программы для управления устройствами компьютера), некоторые другие виды программ [10].

Программы на языках высокого уровня близки к естественному (английскому) языку и представляют набор заданных команд.

Наиболее известные системы программирования:

1. Фортран (FORmula TRANslating system - система трансляции формул); старейший и по сей день активно используемый в решении задач математической ориентации язык.

2. Бейсик (Beginner's All-purpose Symbolic Instruction Code - универсальный символический код инструкций для начинающих); несмотря на многие недостатки и изобилие плохо совместимых версий - самый популярный по числу пользователей [8].

3. Алгол (ALGOrithmic Language - алгоритмический язык); сыграл большую роль в теории, но для практического программирования сейчас почти не используется.

4. ПЛ/1 (PL/I Programming Language - язык программирования первый). Многоцелевой язык; сейчас почти не используется.

5. Си (С - «си»); широко используется при создании системного программного обеспечения [9].

6. Паскаль (Pascal - назван в честь ученого Блеза Паскаля); чрезвычайно популярен как при изучении программирования, так и среди профессионалов. На его базе созданы несколько более мощных языков (Модула, Ада, Дельфи) [7].

7. Кобол (COmmon Business Oriented Language - язык, ориентированный на общий бизнес); в значительной мере вышел из употребления.

8. Дельфи (Delphi) - язык объектно-ориентированного «визуального» программирования; в данный момент чрезвычайно популярен.

9. Джава (Java) - платформенно-независимый язык объектно-ориентированного программирования, чрезвычайно эффективен для создания интерактивных веб-страниц [6].

Среди непроцедурных языков наиболее известны: Лисп (Lisp); Пролог (PROgramming in LOGic); Оккам (назван в честь философа У. Оккама).

Широкое распространение среди разработчиков программ, а также при обучении программированию, получили системы программирования «Турбо» (Turbo) фирмы Borland, ядром которых являются трансляторы с языков программирования Бейсик, Паскаль, Си, Пролог и др. Интерфейс Турбо-оболочки для любых систем программирования внешне совершенно одинаков и предоставляет пользователю стандартный набор функций и команд, описанных выше и отображаемых в главном меню системы [7].

**2. Классификация и компоненты систем программирования**

**2.1 Классификация систем программирования**

По набору входных языков различают системы программирования одно- и многоязыковые. Отличительная черта многоязыковых систем состоит в том, что отдельные части программы можно составлять на разных языках и с помощью специальных обрабатывающих программ объединять их в готовую для исполнения на ЭВМ программу [3].

По структуре, уровню формализации входного языка и целевому назначению различают системы программирования машинно-ориентированные и машинно-независимые.

**2.1.1 Машинно-ориентированные системы программирования**

Данные системы имеют входной язык, наборы операторов и изобразительные средства которых существенно зависят от особенностей ЭВМ (внутреннего языка, структуры памяти и т.д.). Машинно-ориентированные системы позволяют использовать все возможности и особенности машинно-зависимых языков:

* высокое качество создаваемых программ;
* возможность использования конкретных аппаратных ресурсов;
* предсказуемость объектного кода и заказов памяти;
* для составления эффективных программ необходимо знать систему команд и особенности функционирования данной ЭВМ;
* трудоемкость процесса составления программ (особенно на машинных языках и ЯСК), плохо защищенного от появления ошибок;
* низкая скорость программирования;
* невозможность непосредственного использования программ, составленных на этих языках, на ЭВМ других типов [1].

Машинно-ориентированные системы по степени автоматического программирования подразделяются на классы:

1. Машинный язык. В таких системах программирования отдельный компьютер имеет свой определенный Машинный Язык (далее МЯ), ему предписывают выполнение указываемых операций над определяемыми ими операндами, поэтому МЯ является командным. Однако, некоторые семейства ЭВМ (например, ЕС ЭВМ, IBM/370/ и др.) имеют единый МЯ для ЭВМ разной мощности. В команде любого из них сообщается информация о местонахождении операндов и типе выполняемой операции. В новых моделях ЭВМ намечается тенденция к повышению внутренних языков машинно-аппаратным путем реализовывать более сложные команды, приближающиеся по своим функциональным действиям к операторам алгоритмических языков программирования [3].

2. Система Символического Кодирования. В данных системах используются Языки Символического Кодирования (далее ЯСК), которые так же, как и МЯ, являются командными. Однако коды операций и адреса в машинных командах, представляющие собой последовательность двоичных (во внутреннем коде) или восьмеричных (часто используемых при написании программ) цифр, в ЯСК заменены символами (идентификаторами), форма написания которых помогает программисту легче запоминать смысловое содержание операции. Это обеспечивает существенное уменьшение числа ошибок при составлении программ. Использование символических адресов – первый шаг к созданию ЯСК. Команды ЭВМ вместо истинных (физических) адресов содержат символические адреса. По результатам составленной программы определяется требуемое количество ячеек для хранения исходных промежуточных и результирующих значений. Назначение адресов, выполняемое отдельно от составления программы в символических адресах, может проводиться менее квалифицированным программистом или специальной программой, что в значительной степени облегчает труд программиста [10].

3. Автокоды. Существуют системы программирования, использующие языки, которые включают в себя все возможности ЯСК, посредством расширенного введения макрокоманд – они называются Автокоды. В различных программах встречаются некоторые достаточно часто использующиеся командные последовательности, которые соответствуют определенным процедурам преобразования информации. Эффективная реализация таких процедур обеспечивается оформлением их в виде специальных макрокоманд и включением последних в язык программирования, доступный программисту. Макрокоманды переводятся в машинные команды двумя путями – расстановкой и генерированием. В постановочной системе содержатся «остовы» – серии команд, реализующие требуемую функцию, обозначенную макрокомандой. Макрокоманды обеспечивают передачу фактических параметров, которые в процессе трансляции вставляются в «остов» программы, превращая её в реальную машинную программу. В системе с генерацией имеются специальные программы, анализирующие макрокоманду, которые определяют, какую функцию необходимо выполнить и формируют необходимую последовательность команд, реализующих данную функцию. Обе указанных системы используют трансляторы с ЯСК и набор макрокоманд, которые также являются операторами автокода. Развитые автокоды получили название Ассемблеры. Сервисные программы и пр., как правило, составлены на языках типа Ассемблер [3].

4. Макрос. В таких системах язык, являющийся средством для замены последовательности символов описывающих выполнение требуемых действий ЭВМ на более сжатую форму – называется Макрос (средство замены). В основном, Макрос предназначен для того, чтобы сократить запись исходной программы. Компонент программного обеспечения, обеспечивающий функционирование макросов, называется макропроцессором. На макропроцессор поступает макросопределяющий и исходный текст. Реакция макропроцессора на вызов – выдача выходного текста. Макрос одинаково может работать, как с программами, так и с данными [1].

**2.1.2 Машинно-независимые системы программирования**

Машинно-независимые системы программирования – это средство описания алгоритмов решения задач и информации, подлежащей обработке. Они удобны в использовании для широкого круга пользователей и не требуют от них знания особенностей организации функционирования ЭВМ. В таких системах программы, составляемые языках, имеющих название высокоуровневых языков программирования, представляют собой последовательности операторов, структурированные согласно правилам рассматривания языка (задачи, сегменты, блоки и т.д.). Операторы языка описывают действия, которые должна выполнять система после трансляции программы на машинном языке. Таким образом, командные последовательности (процедуры, подпрограммы), часто используемые в машинных программах, представлены в высокоуровневых языках отдельными операторами. Программист получил возможность не расписывать в деталях вычислительный процесс на уровне машинных команд, а сосредоточиться на основных особенностях алгоритма [3].

Среди машинно-независимых систем программирования следует выделить:

1. Процедурно-ориентированные системы. Входные языки программирования в таких системах служат для записи алгоритмов (процедур) обработки информации, характерных для решения задач определенного класса. Эти языки, должны обеспечить программиста средствами, позволяющими коротко и четко формулировать задачу и получать результаты в требуемой форме. Процедурных языков очень много, например: Фортран, Алгол – языки, созданные для решения математических задач; Simula, Слэнг - для моделирования; Лисп, Снобол – для работы со списочными структурами [4].

2. Проблемно-ориентированные системы в качестве входного языка используют язык программирования с проблемной ориентацией. С расширением областей применения вычислительной техники возникла необходимость формализовать представление постановки и решение новых классов задач. Необходимо было создать такие языки программирования, которые, используя в данной области обозначения и терминологию, позволили бы описывать требуемые алгоритмы решения для поставленных задач. Эти языки, ориентированные на решение определенных проблем, должны обеспечить программиста средствами, позволяющими коротко и четко формулировать задачу и получать результаты в требуемой форме. Программы, составленные на основе этих языков программирования, записаны в терминах решаемой задачи и реализуются выполнением соответствующих процедур.

3. Диалоговые системы. Программные средства, обеспечивают оперативное взаимодействие человека с ЭВМ. Обеспечивается оперативное воздействия на прохождение задач. Необходимость обеспечения оперативного взаимодействия с пользователем потребовала сохранения в памяти ЭВМ копии исходной программы даже после получения объектной программы в машинных кодах. При внесении изменений в программу система программирования с помощью специальных таблиц устанавливает взаимосвязь структур исходной и объектной программ. Это позволяет осуществить требуемые редакционные изменения в объектной программе.

4. Непроцедурные системы. Данные обрабатываются по фиксированным алгоритмам (табличные языки и генераторы отчетов). Позволяя четко описывать как задачу, так и необходимые для её решения действия, таблицы решений дают возможность в наглядной форме определить, какие условия должны выполнятся, прежде чем переходить к какому-либо действию. Одна таблица решений, описывающая некоторую ситуацию, содержит все возможные блок-схемы реализаций алгоритмов решения. Табличные методы легко осваиваются специалистами любых профессий. Программы, составленные на табличном языке, удобно описывают сложные ситуации, возникающие при системном анализе [4].

**2.2 Средства создания программ**

Обычно в состав систем программирования входят: трансляторы с языков высокого уровня; средства редактирования, компоновки и загрузки программ; макроассемблеры (машинно-ориентированные языки); отладчики машинных программ [10].

Системы программирования, как правило, включают в себя:

- текстовый редактор (Edit), осуществляющий функции записи и редактирования исходного текста программы;

-загрузчик программ (Load), позволяющий выбрать из директория нужный текстовый файл программы;

- запускатель программ (Run), осуществляющий процесс выполнения программы;

- компилятор (Compile), предназначенный для компиляции или интерпретации исходного текста программы в машинный код с диагностикой синтаксических и семантических (логических) ошибок;

- отладчик (Debug), выполняющий сервисные функции по отладке и тестированию программы;

- диспетчер файлов (File), предоставляющий возможность выполнять операции с файлами: сохранение, поиск, уничтожение и т.п. [5].

В самом общем случае для создания программы на выбранном языке программирования нужно иметь следующие компоненты:

1. Текстовый редактор. Так как текст программы записывается с помощью ключевых слов, обычно происходящих от слов английского языка, и набора стандартных символов для записи всевозможных операций, то формировать этот текст можно в любом редакторе, получая в итоге текстовый файл с исходным текстом программы. Лучше использовать специализированные редакторы, которые ориентированы на конкретный язык программирования и позволяют в процессе ввода текста выделять ключевые слова и идентификаторы разными цветами и шрифтами. Подобные редакторы созданы для всех популярных языков и дополнительно могут автоматически проверять правильность синтаксиса программы непосредственно во время ее ввода.

2. Исходный текст с помощью программы-компилятора переводится в машинный код. Исходный текст программы состоит, как правило, из нескольких модулей (файлов с исходными текстами). Каждый модуль компилируется в отдельный файл с объектным кодом, которые затем требуется объединить в одно целое. Кроме того, системы программирования, как правило, включают в себя библиотеки стандартных подпрограмм (имеют расширение LIB). Стандартные подпрограммы имеют единую форму обращения, что создает возможности автоматического включения таких подпрограмм в вызывающую программу и настройки их параметров.

3. Объектный код модулей и подключенные к нему стандартные функции обрабатывает специальная программа – редактор связей. Данная программа объединяет объектные коды с учетом требований операционной системы и формирует на выходе работоспособное приложение – исполнимый код для конкретной платформы. Исполнимый код это законченная программа, которую можно запустить на любом компьютер, где установлена операционная система, для которой эта программа создавалась. Как правило, итоговый файл имеет расширение .exe или .com.

4. В современных системах программирования имеется еще один компонент – отладчик, который позволяет анализировать работу программы во время ее исполнения. С его помощью можно последовательно выполнять отдельные операторы исходного текста последовательно, наблюдая при этом, как меняются значения различных переменных.

5. В последние несколько лет в программировании (особенно для операционной среды Windows) наметился так называемый визуальный подход. Этот процесс автоматизирован в средах быстрого проектирования. При этом используются готовые визуальные компоненты, свойства и поведение которых настраиваются с помощью специальных редакторов [3].

**Заключение**

Итак, системы программирования - это комплекс инструментальных программных средств, предназначенный для работы с программами на одном из языков программирования.

Системы программирования предоставляют сервисные возможности программистам для разработки их собственных компьютерных программ.

Сегодня имеется немало систем программирования, выпускаемых различными фирмами и ориентированных на различные модели ПК и операционные системы. Наиболее популярны следующие визуальные среды быстрого проектирования: Microsoft Visual Basic, Turbo Pascal, C++ Builder, Symantec Café.

Ключевые понятия систем программирования: язык и реализация языка.

Из универсальных языков программирования сегодня наиболее популярны следующие:

Бейсик (Basic) – для освоения требует начальной подготовки (общеобразовательные школы)

Паскаль (Pascal) – требует специальной подготовки (школы с углубленным изучением предмета и общетехнические вузы)

Си++ (C++), Ява (Java) – требуют профессиональной подготовки (специализированные средние и высшие учебные заведения)

Именно эти системы и языки программирования в дальнейшем будут определять развитие информатики.

Итак, рассмотрев понятие и функции системы программирования, описав существующие языки программирования, классификации и средства создания программ, можем сделать вывод о достижении поставленной цели.

**Список использованных источников**

* 1. Вычислительная техника и программирование / А.В. Петров [и др]. ; под ред. А.В. Петрова. - М.: Высшая школа, 1999. - 258с.
  2. Гейн, А.Г. Основы информатики и вычислительной техники: учебное пособие/ А.Г. Гейн. - М.: Просвещение, 2003. - 246с.
  3. Демьяненко, В.Ю. Программные средства создания и ведения баз данных/ В.Ю.Демьяненко. - М.: Финансы и статистика, 1998. - 216с.
  4. Информатика: Базовый курс/ С.В.Симонович [и др]. - СПб.: Питер, 2001. - 640 с.
  5. Ляхович, В.Ф. Основы информатики: учебное пособие/ В.Ф.Ляхович. - Ростов-на-Дону: Феникс, 2001. - 164с.
  6. Моначов, В. Язык программирования Java и среда NetBeans/ В.Моначев. - 2-е изд. - СПб.: БХВ-Петербург, 2009. - 720 с.
  7. Моргун, А.Н. Справочник по Turbo Pascal для студентов/ А.Н.Моргун. - М.: Диалектика, 2006. - 608 с.
  8. Сайлер, Б. Использование Visual Basic 6. Классическое издание/ Б.Сайлер, Д.Споттс. - М.: Вильямс, 2007. - 832 с.
  9. Страуструп, Б. Язык программирования С++ = The C++ Programming Language / Пер. с англ. - 3-е изд. - СПб.; М.: Невский диалект - Бином, 1999. - 991 с.
  10. Угринович, Н. Информатика и информационные технологии: учебник для 10 - 11 классов/ Н.Угринович. - М.: Бином. Лаборатория знаний, 2007. - 511с.