Министерство образования и науки Российской Федерации

ФГБОУ ВО «Брянский государственный технический университет»

Факультет информационных технологий

Кафедра «Компьютерные технологии и системы»

**О Т Ч Е Т**

по учебной практике (ознакомительная)

Выполнил:
студент группы О-21-ИАС-аид-С
направление подготовки 10.05.04
«Информационно-аналитические системы безопасности», профиль «Автоматизация информационно-аналитической деятельности»
ФИО: Голубев А.В.
«___»_____2022г.
_____
*Подпись студента*

Руководитель практики от БГТУ:
д.т.н., профессор Аверченков В.И.
«___»_____2022г.
_____
*Подпись руководителя*

Оценка отчета:_____
«____»_____2022г.

Брянск 2022

Индивидуальное задание по учебной практике

Студента: Голубева А.В.

Направление подготовки: 10.05.04 - «Информационно-аналитические системы безопасности», профиль «Автоматизация информационно-аналитической деятельности»

Сроки прохождения практики: 28.06.2022г. – 07.06.2022г.

Место прохождения практики: ФГБОУ ВО «Брянский государственный технический университет»

Задание 1. Написание реферата по теме: «Языки и системы программирования».

Задание 2. Создание презентации на основе реферата.

Задание 3. Создание части программы «21 ИАС», содержащий текст реферата.

Студент_____Голубев А.В.
Руководитель производственной практики от кафедры
д.т.н., профессор _____Аверченков В.И.

Министерство образования и науки Российской Федерации

ФГБОУ ВО «Брянский государственный технический университет»

Факультет информационных технологий

Кафедра «Компьютерные технологии и системы»

# ДНЕВНИК
учебной практики студента

Брянск 2022

студент Голубев Артур Владимирович
(Ф.И.О.)

курс 1 направление подготовки 10.05.04 - «Информационно-аналитические системы безопасности»

Направляется на (в): ФГБОУ ВО «Брянский государственный технический университет»

## 1. Календарные сроки практики

По учебному плану начало 28.06.2022г. конец 11.07 2022г.

Дата прибытия на практику «28» июня 2022г.

Дата выбытия с места практики «11» июля 2022г.

## 2. Руководитель практики от БГТУ

Кафедра «Компьютерные технологии и системы»

Ученое звание д.т.н., доцент кафедры КТС

Фамилия Аверченков

Имя Владимир

Отчество Иванович

| Дата | Описание работы, выполненной студентом | Отметка руководителя от базы практики |
|---|---|---|
| 28.06.2022 | Получение задания, ознакомление с темой. | Выполнено |
| 29.06.2022 | Предварительный сбор информации. | Выполнено |
| 30.06.2022 | Начало написания реферата по заданной теме. | Выполнено |
| 01.07.2022 | Написание реферата. | Выполнено |
| 04.07.2022 | Подбор и оформление в список использованной литературы. | Выполнено |
| 04.07.2022 | Оформление текста реферата. | Выполнено |
| 05.07.2022 | Создание презентации на основе реферата. | Выполнено |
| 06.07.2022 | Написание программы на языке C# | Выполнено |
| 07.07.2022 | Написание отчета по выполненной работе | Выполнено |
| 08.07.2022-11.07.2022 | Подготовка к защите отчета. | Выполнено |

Руководитель практики от кафедры
д.т.н., профессор _____/Аверченков В.И.

**График прохождения практики (календарно-тематический план)**

| № | Этапы и разделы практики | Выполняемая работа | Продолжительность в днях |
|---|---|---|---|
| 1. | Ознакомление с темой, составление календарного плана | Занимался поиском информации в сети Интернет | 1 |
| 2. | Поиск литературы и написание реферата | Занимался поиском информации в сети Интернет | 2 |
| 3. | Создание презентации | Создавал презентацию в программе MS PowerPoint | 1 |
| 4. | Написание программы | Писал программу, удовлетворяющую заданию на платформе WPF на языке C# | 3 |
| 5. | Новые знания, умения, практический опыт и составление отчета | Написание отчета | 1 |

Руководитель практики от кафедры
д.т.н., профессор _____/Аверченков В.И.

**Характеристика**

Руководителя учебной практики о работе студента Жукова Д.Ю.

Во время прохождения практики студента ФГБОУ ВО «Брянский государственный технический университет» Голубева А.В. были выполнены такие задачи как: изучение теоретических аспектов по выбранной тематике, ознакомление со спецификой и целями деятельности предприятий, изучение технического и аппаратного обеспечения, выбор цели исследования, а именно создание программы на C#.

В результате прохождения практики, студент Голубев А.В. показал должный уровень теоретической подготовки и ответственности, была показана заинтересованность и стремление к получению новых знаний.

Работа студента Голубева А.В. заслуживает оценки «отлично».

Руководитель практики от кафедры _____/Аверченков В.И.

(подпись) (Фамилия И.О.)

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

Связь между языком, который мы думаем, программой и задачами, и решениями, которые мы можем себе представить, очень близка. Язык предоставляет программисту набор концептуальных инструментов, и если они не соответствуют задаче, то их просто игнорируют. Хороший дизайн и свобода от ошибок не могут быть гарантированы только лингвистическими средствами. Сегодня почти все программы создаются с использованием языков программирования.

Неотъемлемой частью современных компьютеров являются программные системы, являющиеся логическим продолжением логических средств компьютера, и расширяющие возможности аппаратного обеспечения и сферу его применения. Основная цель программного обеспечения — повышение эффективности работы пользователя и увеличение пропускной способности компьютера за счет сокращения времени и затрат на подготовку и выполнение программы. ассемблер

В своей работе я хочу проанализировать несколько современных основных языков, таких как Pascal, Assembler, C++, Basic, Refal и Java, а также системы программирования и их компоненты.

Если проследить историю используемых сегодня языков программирования, таких как C и Pascal (а также менее популярных языков Basic, Fortran или Hell), то окажется, что все они зародились на рубеже 1960-х и 1970-х годов. Другими словами, век современных языков программирования (за исключением Java) прошел уже третье десятилетие, которое является экстремальным периодом для компьютерной индустрии. Современные языки программирования по меньшей мере на десятилетие старше, чем Интернет, Windows и персональные компьютеры. В то же время новые языки не перестали регулярно появляться, но ни один из них не задержался в практике программирования, хотя новые идеи, внедряемые ими, дополняли уже

известные (как это произошло с объектно-ориентированным программированием).

Другой важной особенностью развития языка в последние десятилетия является прекращение попыток создания «универсального» языка программирования, который бы объединил в себе все последние достижения в области развития языка (из экспериментов 1960-х _70-х годов можно вспомнить Algol или Hell). Большие «языковые» проекты, как и языки, создавшие их, безвозвратно уходят в прошлое.

Наконец, появление персонального компьютера и операционной системы с графическим интерфейсом пользователя (в первую очередь, MacOS и Windows) переместило внимание разработчиков программного обеспечения из области языков программирования в другие области средств разработки программного обеспечения, такие как визуальное или объектно-ориентированное программирование, сетевые протоколы или модели баз данных. Сегодня программист использует в качестве инструмента специфическую систему программирования (например, Delphi), а не язык, и не так важно, какой язык является основой.

Таким образом, интерес к языкам программирования снизился, а диапазон используемых языков стабилизировался. В некотором смысле можно предположить, что «все сказано и сделано» в области языков программирования, и разработка средств разработки программного обеспечения будет продолжаться по-другому. Настало время проанализировать современные языки программирования и узнать о достигнутых практических результатах.

# ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Язык программирования — это формальная символическая система для записи компьютерных программ. Язык программирования определяет набор лексических, синтаксических и семантических правил, определяющих внешний вид программы и действия, которые должен выполнять исполнитель (обычно компьютер) под своим управлением.

Языки программирования классифицируются по различным критериям. Однако они обычно делятся на языки высокого и низкого уровня. Чем ближе язык к естественному, тем больше вероятность того, что он будет классифицирован как язык высокого уровня. И наоборот, если язык ближе к машинным инструкциям, его называют языком низкого уровня. Поэтому весь спектр языков программирования можно разделить на две группы: Языки низкого уровня и языки высокого уровня.

## ЯЗЫКИ ПРОГРАММИРОВАНИЯ НИЗКОГО И ВЫСОКОГО УРОВНЯ

Подчиненные языки включают в себя языки ассемблера (от ассемблера до ассемблера). Языки ассемблера используют символы для команд, которые легко понять и запомнить. Вместо последовательности двоичных кодов инструкций записываются их символические имена, а вместо двоичных адресов данных, используемых при выполнении инструкции, — выбранные программистом символические имена этих данных. Иногда язык ассемблера называется Mnemocode или Autocode.

Большинство программистов используют языки высокого уровня для программирования. Подобно обычному человеческому языку, такой язык имеет свой собственный алфавит — много символов, используемых в языке. Эти символы используются для формирования так называемых ключевых слов языка. Каждое из ключевых слов выполняет свою функцию, так же как и в

нашем обычном языке слова, составленные из букв алфавита языка, могут выполнять функции различных частей слов. Ключевые слова соединяются в предложениях в соответствии с определенными синтаксическими правилами языка. Каждое предложение определяет определенную последовательность действий, которые должен выполнять компьютер.

Язык высокого уровня выступает в качестве посредника между человеком и компьютером и позволяет человеку общаться с компьютером более знакомым образом. Часто этот язык помогает в выборе правильного способа решения проблемы.

Перед написанием программы на языке высокого уровня программисту необходимо придумать алгоритм решения задачи, т.е. пошаговый план действий, которые необходимо предпринять для решения задачи. По этой причине языки, требующие предварительной подготовки алгоритма, часто называют алгоритмическими языками.

## ПАСКАЛЬ

В 1968 году Никлаус Вирт написал первый компилятор языка, Паскаль. Этот язык был назван в честь выдающегося французского математика Блеза Паскаля. Паскаль — успешный, общеприменимый язык, который подходит для программирования как научных, так и коммерческих задач. Входные/выходные возможности этого языка несколько слабее, чем у коммерческих языков, таких как COBOL, поэтому он никогда не претендовал на замену. Тем не менее, с момента своего первого релиза Pascal является довольно солидным языком, который успешно выполняет свою работу.

Популярность Паскаля резко возросла в 1970-х годах. Самым большим преимуществом этого языка является поддержка концепции структурного программирования, что позволяет сделать программы более удобными для внесения изменений. Идеология структурного программирования

интегрирована в язык, поэтому программы на Pascal легче поддерживать, чем программы, написанные на других языках того времени.

В 1970-х годах Паскаль был «единственным языком программирования, подходящим для всех». Компания IBM попыталась создать нечто подобное с помощью языка PL/I. Как и ПЛ/И, Паскаль не достиг высшей цели. Популярность языка Паскаль снижалась так же быстро, как и росла. В 1970-х годах наблюдался огромный рост использования Паскаля, а в 1980-х годах интерес к Паскалю резко упал.

Несмотря на потерю позиции, Паскаль открыл путь другим языкам для поддержки структурных концепций, удобства обслуживания и свободного использования программ.

## АССЕМБЛЕР

Язык ассемблера является символическим представлением машинного языка. Это упрощает процесс программирования по сравнению с программированием в машинном коде. Некоторые задачи, такие как обмен сложными структурами с нестандартными устройствами обработки данных, не могут быть решены с помощью языков программирования более высокого уровня. Сборщик может это сделать. В основном, язык ассемблера — это машинный язык. А программист, реализующий задачу на языке высокого уровня с использованием языка ассемблера, может определить, имеет ли смысл с точки зрения использования компьютера решать эту задачу. У ассемблера есть особенность, которая отпугивает многих новичков в языках программирования — ассемблер — это язык, зависящий от машины. Это означает, что ассемблер работает непосредственно с ресурсами компьютера, что требует хорошего знания его архитектуры, логики работы операционной системы и высокой степени точности при написании программы.

Несмотря на то, что ассемблер является машинно-ориентированным языком, т.е. языком низкого уровня, программист может использовать его как для работы на высоком, так и на среднем уровне. Низкий уровень программирования на ассемблере подразумевает прямой доступ к каналам ввода/вывода устройств, называемым идеальными портами, и прямой доступ к оперативной памяти. Использование этого режима позволяет неопытному программисту использовать более четкий и легкий стиль разработки программы. Более опытные программисты ассемблерного языка могут использовать такие идеальные возможности режима, как вложенные структуры и объединения.

Важной особенностью режима Ideal является использование проверок типов данных, аналогичных тем, которые используются в языках высокого уровня, что позволяет выявить множество ошибок при переводе.

Ассемблер предоставляет программисту полную свободу действий при разработке программы, что является как его преимуществом, так и недостатком, так как требует от разработчика знания системы команд данного компьютера и его операционной системы. Более того, несмотря на минимальный размер исполняемого файла на максимальной скорости, время создания программы значительно увеличивается с увеличением объема разрабатываемого проекта. По этой причине ассемблер был и остается языком программирования для профессионалов.

C++

C++ был разработан в начале 1980-х годов шведским программистом по имени Bjarne Stroustrup. В C++ есть несколько дополнительных команд и операторов, но главное отличие — это подход к программированию.

Основная причина, по которой Си++ до сих пор популярен, заключается в том, что он поддерживает объектно-ориентированное программирование

(ООП). ООП — это еще один способ написания программ, помогающий программисту писать программы быстрее и с меньшим количеством ошибок. ООП также позволяет увеличить скорость обслуживания.

Так как C++ в основном основан на C, акцент в C на низкоуровневых инструментах достаточен для решения наиболее актуальных задач системного программирования. В свою очередь, C в значительной степени обязан своим предшественником ППГ. C++ — машинно-ориентированный язык программирования, предназначенный для того, чтобы сделать программирование более приятным для серьезного программиста.

В C++ нет типов данных высокого уровня и нет первичных операций высокого уровня. Например, он не содержит матричного типа с операциями обработки.

Если пользователю нужен этот тип, его можно определить на самом языке. Фактически, основной задачей программирования на C++ является определение универсальных и специальных типов приложений.

Популярность объектно-ориентированного C++ создала много новых языков для современного Интернета. Причиной успешного развития Java (и его производных, таких как JavaScript) является широкое использование C++, и хотя C++ не поддерживает достаточную безопасность при программировании веб-сайтов, его объектно-ориентированная природа делает Java хорошим кандидатом для программирования Интернет-объектов. Компания Sun Microsystems изменила язык C++, добавив элементы с достаточной безопасностью и удалив некоторые сомнительные элементы из языка C++. (Одним из примеров является множественное наследование, которое позволяет генерировать один объект программирования, например, окно, из набора нескольких родительских окон с различными свойствами. Такая концепция запутанна, даже если она сформулирована на высоком уровне).

BASIC

BASIC был разработан в Дартмутском колледже и предназначен для начинающих. ФОРТРАН был сложным языком для студентов не математических факультетов. Джон Кемени и Томас Курц, учившиеся в Дартмуте, использовали FORTRAN в качестве основы для разработки BASIC. BASIC — это аббревиатура от Beginner's All-purpose Symbolic Instruction Code, название, которое говорит само за себя. Интерпретатор обычно используется с BASIC, хотя современные версии BASIC скомпилированы. Это позволяет начинающим программистам сконцентрироваться на языке программирования, а не на специфике компиляции. Как и все интерпретируемые языки, программы на BASIC работают медленнее скомпилированных программ, поэтому программы на BASIC не используются в коммерческих приложениях.

Этот язык стал одним из самых распространенных. BASIC был изначально задуман как очень простой язык, который можно выучить очень быстро. Язык содержал примитивные инструменты редактирования, освобождавшие пользователя от необходимости понимать все сложности базовой операционной системы. Вначале Basic использовался только для обработки числовых значений, позже он был расширен обработкой строковых переменных и теперь обеспечивается рядом методов простой манипуляции через символьные цепочки, которые фактически стали стандартом. Простота Basic сделала его наиболее подходящим языком программирования для первых микрокомпьютеров. Сегодня это основной язык программирования для персональных компьютеров. К сожалению, почти каждая машина имеет свой собственный диалект на BASIC, поэтому обеспечить мобильность программ на нем довольно сложно. В настоящее время многие основные диалекты включают в себя современные структуры управления, такие как: REPEAT… UNTIL, что дает определенный оптимизм.

Одним из языков, разработанных в России, является Refal, который был разработан в СССР в 1966 году. Этот язык прост и удобен для описания манипуляций с любыми текстовыми объектами.

Refal широко используется в разработке переводчиков с алгоритмических языков, как универсальных, так и проблемно-ориентированных, и автокодов. Помимо использования в задачах перевода, Refal имеет такие важные приложения, как машинное выполнение громоздких аналитических вычислений в теоретической физике и прикладной математике; разработка «интеллектуальных» информационных систем, выполняющих нетривиальную логическую обработку информации; математически устойчивые теоремы; моделирование целенаправленного поведения; разработка диалоговых обучающих систем; исследования в области искусственного интеллекта и др.

Программирование на Refal имеет особенность, которая в первую очередь связана с тем, что Refal является языком функционального типа, в отличие от обычных операторских языков, таких как Algol, Fortran и др. Если программа на операторских языках представляет собой не более чем набор операторов команд, то программа на языке Refal по сути является описанием отношений и отношений между определенными операторами. Для определения структур в Refal используются круглые скобки, а особенности всех реализаций языка таковы, что использование круглых скобок резко повышает эффективность выполнения программы.

Переменные типа «выражение» также имеют определенную специфику, т.е. их способность расширяться в процессе идентификации. Правильное использование переменных этого типа также позволяет значительно повысить эффективность программ Refal.

JAVA

Java — объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems. Java-приложения обычно компилируются в специальный байтовый код, так что они могут быть выполнены на любой Java-виртуальной машине (JVM) независимо от архитектуры компьютера. Официальная дата выхода — 23 мая 1995 г. Первоначально язык назывался Oak, он был разработан Джеймсом Гослингом для программирования устройств бытовой электроники. Позже он был переименован в Java и использовался для написания клиентских приложений и серверного программного обеспечения.

Преимущество выполнения программ в полной независимости байт-кода от операционной системы и устройств позволяет Java-приложениям запускаться на любом устройстве, для которого имеется соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности, благодаря тому, что выполнение программы полностью контролируется виртуальной машиной. Любая операция, превышающая определенные разрешения программы (например, попытка получить несанкционированный доступ к данным или подключиться к другому компьютеру), приводит к немедленному прерыванию работы.

Одним из недостатков концепции виртуальной машины является то, что выполнение байтового кода виртуальной машиной может снизить производительность программ и алгоритмов, реализованных на языке Java. Это утверждение было верно для первых версий Java Virtual Machine, но в последнее время оно почти утратило свою значимость. Этому способствовал ряд улучшений: использование технологии трансляции байт-кода в машинный код во время работы программы (технология JIT) с возможностью хранения версий класса в машинном коде, широкое использование платформо-ориентированного (нативного) кода в стандартных библиотеках, аппаратного обеспечения, позволяющего ускорить обработку байт-кода (например,

технология Jazelle, поддерживаемая некоторыми процессорами компании ARM).

Идеи, лежащие в основе концепции и различных реализаций Java Virtual Machine Environment, вдохновили многих энтузиастов на расширение списка языков, которые могут быть использованы для создания программ, запускаемых на виртуальной машине. Эти идеи также нашли выражение в общей спецификации инфраструктуры языка CLI, лежащей в основе платформы Microsoft .NET.

## ВОЗМОЖНОСТИ РЕАЛИЗАЦИИ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

Недостаточно создать язык, подходящий для написания программ. Для каждого языка нужен свой переводчик. Такие переводчики являются специальными переводческими программами.

Транслятор — это программа, которая служит для перевода программы, написанной на одном языке программирования, в программу, написанную на другом языке программирования. Тексты исходных и результирующих программ сохраняются в памяти компьютера. Примером транслятора является компилятор. Компилятор — это транслятор с текста на машинный язык, который читает исходный код. Он оценивает его в соответствии с синтаксической структурой языка и переводит на машинный язык. Другими словами, компилятор не выполняет программы, а собирает их. Процесс такого перевода называется компиляцией.

Байтовый код — это промежуточный подход, при котором программа преобразуется в промежуточную двоичную форму, которая интерпретируется «виртуальной машиной» во время выполнения.

Компилятор выдает полный результат — программу в машинном коде. Затем эта программа выполняется. Вы можете сохранить скомпилированную версию исходной программы на диск. Для повторного запуска исходной

программы компилятор не нужен. Достаточно загрузить скомпилированную ранее версию с жесткого диска в память компьютера и выполнить ее.

Существует еще одна возможность совместить процессы перевода и выполнения программ. Это называется интерпретация. Суть процесса интерпретации заключается в следующем. Сначала она транслируется в машинный код, а затем выполняется первая строка программы. Когда первая строка заканчивается, переводится вторая строка, которая затем выполняется, и так далее. Этот процесс управляется программой-переводчиком.

Переводчик — это программа, предназначенная для поточного перевода и выполнения исходной программы. Этот процесс называется интерпретацией. Другими словами, можно сказать, что интерпретатор моделирует виртуальную вычислительную машину, где основными инструкциями являются не элементарные инструкции процессора, а операторы языка программирования. Процесс перевода включает в себя проверку исходной программы на соответствие правилам используемого языка. При обнаружении ошибок в программе транслятор вводит сообщение о них на выходном устройстве (обычно на экране).

Интерпретатор сообщает об ошибках, найденных после перевода каждой строки программы. Компилятор транслирует программу намного быстрее, чем интерпретатор, но сообщает об обнаруженных ошибках только после компиляции всей программы. В этом случае сложнее находить и исправлять ошибки. Переводчики предназначены для языков, предназначенных для обучения программированию, и используются начинающими программистами. Большинство современных языков предназначены для разработки сложных программных пакетов и компиляции.

Иногда один и тот же язык может использовать и компилятор, и интерпретатор. Такими языками являются, например, Basic. Как правило, компиляторы и интерпретаторы приравниваются к языкам, с которых они должны переводиться. Слова Pascal, Ada, C могут относиться как к названиям языков, так и к названиям соответствующих программ.

Недостатком компилятора является сложность трансляции языков программирования, ориентированных на обработку данных сложных структур, которые часто заранее неизвестны или динамически изменяются в процессе работы программы. С помощью интерпретатора, с другой стороны, можно в любое время остановить программу, изучить содержимое памяти, организовать диалог с пользователем, выполнить сложные преобразования и в то же время постоянно следить за состоянием окружающей программной и аппаратной среды, достигая тем самым высокой степени надежности.

## КЛАССИФИКАЦИЯ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

Существующие языки программирования делятся на четыре основные группы: процедурные, объектно-ориентированные, функциональные и логические. Давайте кратко определим каждый подход.

Процедурное программирование — это такое программирование, при котором программа отделена от данных и состоит из последовательности инструкций, обрабатывающих данные. Данные обычно хранятся в виде переменных. Весь процесс расчета рассчитан на изменение их содержания.

Языки декларативного программирования — это языки для объявлений и построения структур. Сюда входят функциональные и логические языки программирования. Эти языки явно не выполняют алгоритмических действий, то есть алгоритм не задается программистом, а собирается программой. Декларативные языки указывают, строят структуру или систему, то есть объявляют (декларируют) некоторые из атрибутов создаваемого объекта. Эти языки получили широкое применение в системах автоматизированного проектирования (САПР), в так называемых CAD-пакетах, в моделировании, в системах искусственного интеллекта.

Объектно-ориентированное программирование — в этих языках переменные и функции сгруппированы в так называемые классы (шаблоны).

Это обеспечивает более высокий уровень структуризации программы. Объекты, которые возникают из классов, вызывающих методы (функции или процедуры) друг у друга, и таким образом изменяют состояние свойств (переменных). С формальной и математической стороны объектно-ориентированный способ написания программ основан на процедурной модели программирования, но с содержательной стороны ООП основывается не на функции, а на объекте как целостной системе со стандартным автоматическим объектным интерфейсом.

Сетевые языки — языки, предназначенные для организации взаимодействия удалённых компьютеров в интенсивном интерактивном режиме, и поэтому они основаны на принципах интерпретации, т.е. построчной интерактивной обработки строк программного кода, описывающих сценарий (сценарий) сетевого взаимодействия компьютеров, поэтому их часто называют скриптовыми языками, хотя скриптовые языки не обязательно являются сетевыми языками, например, пакетными командными языками различных операционных сред.

## МАШИННО-ОРИЕНТИРОВАННЫЕ ЯЗЫКИ

Машинно-ориентированные языки — это языки, набор операторов и их графических средств которых существенно зависит от свойств компьютера (внутренний язык, структура памяти и т.д.). Машинно-ориентированные языки позволяют использовать все возможности и характеристики машинно-зависимых языков: высокое качество создаваемых программ (компактность и скорость исполнения); возможность использования специфических аппаратных ресурсов; предсказуемость объектного кода и порядка памяти; для использования машинно-ориентированных языков необходимо иметь высокую степень гибкости. Для создания эффективных программ необходимо знать систему управления и свойства функциональности данного компьютера;

сложность процесса создания программ (особенно на машинных языках и NASK), плохо защищенных от возникновения ошибок; низкая скорость работы программ (особенно на компьютере). Машинно-ориентированные языки делятся на классы в зависимости от степени автоматизации программирования.

### Машинный язык

Одиночный компьютер имеет свой специфический машинный язык (далее ML), ему предписано выполнять заданные операции над операндами, которые он определяет, поэтому ML является командным языком. Однако некоторые семейства компьютеров (например, ЕС, IBM/370/ и т.д.) имеют один ТЕ для компьютеров различной производительности. Инструкция каждого из них содержит информацию о расположении операндов и типе выполняемой операции.

В новых компьютерных модулях наблюдается тенденция к увеличению внутренних языков машины и оборудования для реализации более сложных инструкций, где операторы алгоритмических языков программирования обращаются своими функциональными действиями.

### Языки символьного кодирования

Языки символьного кодирования (далее — «языки кодирования») являются командными языками, так же, как и МЕ. Однако, коды операций и адреса в машинных инструкциях, представляющие собой последовательность двоичных (внутренний код) или восьмеричных цифр (часто используемых при написании программ), заменяются в СНБ символами (идентификаторами), форма написания которых помогает программисту легче запомнить семантическое содержание операции. Это приводит к значительному сокращению количества ошибок при написании программ.

## Автокоды

Существуют также языки, которые содержат все возможности NSC за счет расширенного введения макрокоманд — они называются Autocodes. В различных программах существует несколько достаточно общих последовательностей команд, которые соответствуют определенным процедурам преобразования информации. Эффективная реализация таких процедур обеспечивается их формализацией в виде специальных макроинструкций и интеграцией в язык программирования, доступный программисту. Макроинструкции переводятся в машинные инструкции двумя способами — упорядочивая и генерируя. Система инсценировки содержит «скелеты» — серию инструкций, реализующих требуемую функцию, заданную командой макроса.

Макроинструкции обеспечивают передачу фактических параметров, которые вставляются в «скелет» программы во время перевода, что делает ее настоящей машинной программой. В системе с генерацией существуют специальные программы, которые анализируют макроинструкции, определяют, какая функция должна быть выполнена, и формируют необходимую последовательность инструкций, реализующих эту функцию. В обеих системах используются трансляторы NSC и набор макроинструкций, которые также являются операторами автокодирования. Разработанные автокоды называются ассемблерами. Утилиты и т.д. обычно компилируются на таких языках, как ассемблер. Подробнее о языке ассемблера я расскажу в главе

## Macros

Язык, который является средством для замены строки, описывающей требуемые действия компьютера, на более сжатую форму, называется макросом (замена). В принципе, макрос предназначен для сокращения записи исходной программы. Программный компонент, обеспечивающий работу макросов, называется макропроцессором. Процессор макросов получает

определение макроса и исходный код. Макрос может работать как с программами, так и с данными одинаково.

## МАШИННЫЕ НЕЗАВИСИМЫЕ ЯЗЫКИ

Машинно-независимые языки — это средства описания алгоритмов решения проблем и обрабатываемой информации. Они удобны в использовании для широкого круга пользователей и не требуют от них знания специфики организации функционирования компьютеров и БК.

Такие языки называются языками программирования высокого уровня. Программы, написанные на таких языках, являются операторными последовательностями, структурированными по правилам анализа языка (задачи, сегменты, блоки и т.д.). Языковые операторы описывают действия, которые система должна выполнять после трансляции программы на MJ. Это означает, что последовательности команд (процедуры, подпрограммы), часто используемые в машинных программах, представлены на языках высокого уровня отдельными операторами. Программисту была предоставлена возможность не подробно описать процесс вычисления на уровне машинных инструкций, а сконцентрироваться на существенных особенностях алгоритма.

### Проблемно-ориентированные языки

С расширением областей применения компьютерных технологий возникла необходимость формализации представления постановки и решения новых классов задач. Необходимо было создать такие языки программирования, которые, используя термины и терминологию в этой области, позволяли бы описывать необходимые алгоритмы решения задач, они становились проблемно-ориентированными языками. Эти языки — языки, ориентированные на решение определенных задач, должны дать

программисту средства, позволяющие кратко и точно сформулировать задачу и получить результат в требуемой форме.

Например, существует много проблемных языков: Fortran, Algol — языки, созданные для решения математических задач; Simula, Slang — для моделирования; Lisp, Snobol — для работы со списочными структурами.

### Универсальные языки

Для широкого круга задач были созданы универсальные языки: торговля, наука, моделирование и т.д. Первый универсальный язык был разработан компанией IBM, которая стала Pl/1 в порядке языков. Вторым по мощности универсальным языком является алголь-68, который позволяет работать с символами, битами, числами с фиксированными и плавающими точками. Pl/1 имеет развитую систему операторов для управления форматами, работы с полями переменной длины, с данными, организованными в сложных структурах и эффективного использования каналов связи. Язык учитывает возможности прерывания, которые есть во многих машинах, и имеет соответствующих операторов. Предусмотрена возможность параллельного выполнения разделов программы. Программы в Pl/1 компилируются с использованием автоматических процедур. Язык использует многие свойства Фортрана, Алголя, Кобола. Но это позволяет не только динамическое, но и управляемое и статистическое выделение памяти.

### Языки диалога

Появление новых технических навыков поставило перед системными программистами задачу создания программных средств, обеспечивающих быстрое взаимодействие человека с компьютером, их назвали диалоговыми языками. Эта работа была выполнена в двух направлениях. Были созданы специальные языки управления, позволяющие оказывать оперативное влияние на передачу задач, выполняемых на ранее неразвитом (недиалоговом) языке.

Были также разработаны языки, которые, помимо контрольных целей, должны позволять описывать алгоритмы решения проблем.

Необходимость обеспечения быстрого взаимодействия с пользователем требовала сохранения копии исходной программы в памяти компьютера даже после того, как объектная программа была получена в машинном коде. При внесении изменений в программу с помощью языка диалогов система программирования использует специальные таблицы для установления связи между структурами исходной и объектной программ. Это позволяет вносить необходимые редакторские изменения в объектную программу.

Примером языков общения является Basic. В нем используются нотации, сходные с обычными математическими выражениями. Многие операторы являются упрощенными версиями операторов Fortran. Таким образом, этот язык позволяет решать достаточно широкий круг задач.

### Непроцедурные языки

Непроцедурные языки — это группа языков, описывающих организацию данных, обрабатываемых с использованием фиксированных алгоритмов (языки таблиц и генераторы отчетов), а также языки для связи с операционными системами. Предоставляя четкое описание как задачи, так и действий, необходимых для ее решения, таблицы решений обеспечивают средства для четкого определения условий, которые должны быть выполнены до того, как может быть инициировано то или иное действие. Таблица решений, описывающая конкретную ситуацию, содержит все возможные блок-схемы для реализации алгоритмов решения.

Табличные методы легко усваиваются профессионалами всех профессий. Программы, написанные на табличном языке, удобно описывают сложные ситуации, возникающие при анализе системы.

# ЗАКЛЮЧЕНИЕ

Изобретение самого высокого языка программирования позволило нам общаться с машиной, понимать ее (конечно, только если ты знаешь используемый язык). Если посмотреть на темпы роста и развитие новейших технологий в программировании, то можно предположить, что в ближайшем будущем человеческие знания в этой области помогут создать языки мира, которые смогут получать, обрабатывать и передавать информацию в виде мыслей, слов, звуков или жестов.

Современные языки программирования обладают огромными преимуществами по сравнению с более ранними языками. Они более структурированы и предлагают интегрированную среду развития.

В 1970-х годах Паскаль был самым популярным языком в общем употреблении, но в 1980-х годах его использование резко сократилось.

Язык C, разработанный Bell Laboratories, является очень эффективным, но низкоуровневым языком программирования. C — это основа современных языков.

Самым большим преимуществом C++ перед его предшественником является поддержка объектно-ориентированного программирования.

Visual Basic, преемник языка BASIC, предлагает лучшую базу программирования для начинающих.

Выбор используемого языка определяется многими факторами.

Большинство языков имеют специализацию и подходят для написания определенных типов программ. Выбор языка определяется ориентацией разрабатываемой программы. Кроме того, программист должен знать, насколько распространен этот язык в случае, если кому-то понадобится поддерживать его программу в будущем.

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ


1. «Информационные технологии и ИКТ. 10-11», Н.Д. Угринович, Москва, 2006;
2. Технологии управления информацией»: Инфоматика-М.Ф. Меняев: Омега-Л, 2003;
3. «Информатика», базовый курс, 2-е издание / Под руководством С.В. Симоновича-СПб. Питер, 2004;
4. Леонтьев. Последняя энциклопедия компьютерного программного обеспечения. — Пресса, 2003;
5. Ассемблер для Windows, Пирогов В.Ю. 2005 ;
6. Магданурова Г. I, «Визуальные основы на практике», 2004..;
7. Васильев П. П. «Турбо Паскаль в примерах и задачах», 2003;
8. Есипов А.С., «Информатика и информационные технологии», 2003;
9. В. В. Васильченко «ФОРТРАН». Программирование Windows-приложений на FORTRAN. «», 2003 г;
10. Брюс Экель «Java Philosophy», 2014.