# Реферат

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ - совокупность программ системы обработки информации и программных документов, необходимых для эксплуатации этих программ (ГОСТ 19781-90). Также — совокупность программ, процедур и правил, а также документации, относящихся к функционированию системы обработки данных (СТ ИСО 2382/1-84).

ИНСТРУМЕНТАЛЬНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ – программное обеспечение, предназначенное для использования в ходе проектирования, разработки и сопровождения программ. Обычно этот термин применяется для акцентирования отличия данного класса ПО от прикладного и системного программного обеспечения.

КОМПИЛЯТОР – транслятор, выполняющий преобразование программы, составленной на исходном языке, в объектный модуль.

ИНТЕРПРЕТАТОР – программа (иногда аппаратное средство), анализирующая команды или операторы программы и тут же выполняющая их.

ОПЕРАЦИОННАЯ СИСТЕМА - комплекс управляющих и обрабатывающих программ, которые, с одной стороны, выступают как интерфейс между устройствами вычислительной системы и прикладными программами, а с другой — предназначены для управления устройствами, управления вычислительными процессами, эффективного распределения вычислительных ресурсов между вычислительными процессами и организации надёжных вычислений.

ПРИКЛАДНАЯ ПРОГРАММА - программа, предназначенная для выполнения определенных пользовательских задач и рассчитанная на непосредственное взаимодействие с пользователем.

VISUAL BASIC - средство разработки программного обеспечения, разрабатываемое корпорацией Microsoft и включающее язык программирования и среду разработки.

VISUAL BASIC FOR APPLICATION - немного упрощённая реализация языка программирования Visual Basic, встроенная в линейку продуктов Microsoft Office (включая версии для Mac OS), а также во многие другие программные пакеты, такие как AutoCAD, SolidWorks, CorelDRAW, WordPerfect и ESRI ArcGIS.

Цель работы – исследование видов, функций программного обеспечения, в частности, инструментального.

Классификация программного обеспечения:

1. Прикладные программы, непосредственно обеспечивающие выполнение необходимых пользователям работ;
2. системные программы, предназначены для управления работой вычислительной системы, выполняют различные вспомогательные функции, например:
3. инструментальные программные системы, облегчающие процесс создания новых программ для компьютера.

# Виды инструментального программного обеспечения:

1. Текстовые редакторы
2. Интегрированные среды разработки
3. SDK
4. Компиляторы
5. Интерпретаторы
6. Линковщики
7. Парсеры и генераторы парсеров (см. Javacc)
8. Ассемблеры
9. Отладчики
10. Профилировщики
11. Генераторы документации
12. Средства анализа покрытия кода
13. Средства непрерывной интеграции
14. Средства автоматизированного тестирования
15. Системы управления версиями и др.

Для создания программы на выбранном языке программирования нужно иметь следующие компоненты:

1. Текстовый редактор для создания файла с исходным текстом программы.

2. Компилятор или интерпретатор. Исходный текст с помощью программы-компилятора переводится в промежуточный объектный код.

3. Редактор связей или сборщик, который выполняет связывание объектных модулей и формирует на выходе работоспособное приложение - исполнимый код.

Результат работы: Рассмотрено программное обеспечение, его функции и виды, в частности инструментальное программное обеспечение, его сущность, задачи. В третьей главе рассмотрен Microsoft Visual Basic как средство разработки программного обеспечения и его диалект - Microsoft Visual Basic for Application. В курсовой работе реализован алгоритм решения финансово-экономической задачи с использованием языка программирования Pascal.

# Введение

В современном мире уже не один человек, попробовавший блага цивилизации, не может представить свою жизнь без использования компьютерной техники. Ее использование происходит в любой сфере человеческой жизнедеятельности: производстве, торговле, обучении, развлечении и общении людей, их научной и культурной деятельности. Все это благодаря возможности подбора компьютерной техники для решения любой, даже самой сложной задачи.

Однако и универсальность и специализированность компьютерной техники обеспечивается использованием на базе практически любой ЭВМ различного набора программного обеспечения, обеспечивающих решение любых поставленных задач.

Все мы видим огромное многообразие компьютерных программ и ошеломляющие темпы их роста и совершенствования, и лишь малая часть из нас представляет невидимую сторону по их проектированию, разработке и созданию. Однако данная сфера компьютерных технологий является на наш взгляд наиболее важной, так как именно от ее развития будет зависеть бедующее компьютерных технологий.

А так как разработка любой компьютерной программы происходит с использованием Инструментального программного обеспечения, то в нашей курсовой работе хотелось бы подробно остановиться именно на нем, выделив его из всего программного обеспечения и раскрыв его сущность и особенности.

Для наглядности мы рассмотрим инструментальное программное обеспечение (объект исследования) на примере программного комплекса Visual Basic for Application (предмет исследования), применяемого для программирования в среде Microsoft Office – самого распространенного и популярного офисного пакета.

# Программное обеспечение

## 1.1 Понятие и сущность программного обеспечения

Программное обеспечение (ПО) – неотъемлемая часть компьютерной системы. Оно является логическим продолжением технических средств любого компьютера. Сфера применения конкретного компьютера определяется созданным для него ПО. Сам по себе компьютер не обладает знаниями ни в одной области применения. Все эти знания сосредоточены в выполняемых на компьютерах программах, которые имеют набор определенных функциональных возможностей и предназначены для выполнения конкретных, в большинстве случаев, узкоспециализированных функций, таких например как создание и обработка графических изображений или звуковых файлов.

Программное обеспечение в настоящее время составляет сотни тысяч программ, которые предназначены для обработки самой разнообразной информации с самыми различными целями.

К программному обеспечению (ПО) относится также вся область деятельности по проектированию и разработке ПО:

1. технология проектирования программ (например, нисходящее проектирование, структурное и объектно-ориентированное проектирование);
2. методы тестирования программ;
3. методы доказательства правильности программ;
4. анализ качества работы программ;
5. документирование программ;
6. разработка и использование программных средств, облегчающих процесс проектирования программного обеспечения, и многое другое.

Существует множество различных определений ПО. Вообще, программное обеспечение — совокупность программ системы обработки информации и программных документов, необходимых для эксплуатации этих программ (ГОСТ 19781-90). Также — совокупность программ, процедур и правил, а также документации, относящихся к функционированию системы обработки данных (СТ ИСО 2382/1-84).

Программное обеспечение является одним из видов обеспечения вычислительной системы, наряду с техническим (аппаратным), математическим, информационным, лингвистическим, организационным и методическим обеспечением.

В компьютерном сленге часто используется слово софт от английского слова software, которое в этом смысле впервые применил в статье в American Mathematical Monthly математик из Принстонского университета Джон Тьюки (англ. John W. Tukey) в 1958 году.

Другие определения:

1. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ — это совокупность программ, позволяющих осуществить на компьютере автоматизированную обработку информации.
2. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ (математическое **обеспечение** электронной вычислительной машины), совокупность программ системы обработки данных и **программных** документов, необходимых для реализации программ на электронной вычислительной машине.
3. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ — совокупность программ для управления процессом работы компьютера, автоматизации программирования.
4. **ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ** — комплекс компьютерных программ, обеспечивающий обработку или передачу данных.

Все определения похожи и отражают суть программного обеспечения – организация взаимодействия аппаратной (технической) части, в виде различных встроенных узлов и периферических устройств, их контроль и координация общего взаимодействия компьютерной системы между собою и с пользователем.

## 1.2 Функции программного обеспечения

Приведенные выше понятия программного обеспечения обуславливают функции, выполняемые программным обеспечением в процессе функционирования компьютерной техники. Перечень данных функций весьма разнообразен, но условно их можно разбить на следующие пять типов:

1. Аппаратно-механические. Осуществляют сопряжение различных компонентов компьютера, обеспечивают передачу аппаратного сигнала от одного компонента к другому.

2. Машинно-Логические. Обрабатывают и интерпретируют набор электромагнитных импульсов аппаратного обеспечения в логически осознанный программный код, обладающий определенной структурой и свойствами.

3. Информационно-командные. Осуществляют проверку соответствия программного кода принципам системы и создание логической структуры информации и осуществляют его исполнение.

* 1. 4. Интерфейсные. Обеспечивают обработку и интерпретацию программного кода в формат отображения доступный для восприятия пользователем. Создает благоприятную среду для взаимодействия «Компьютер-Человек, Человек-Компьютер».

5. Прикладные. Осуществляет математические, логические, физические и другие действия с набором имеющихся данных, другими словами обработку имеющейся информации для решения определенных задач.

Данный перечень является далеко не исчерпывающим, что говорит о многообразии и неоднозначности функций, выполняемых программным обеспечением.

## 1.3 Виды программного обеспечения

В зависимости от функций, обеспечиваемым определенным компонентом компьютера, возникает необходимость создания для него своего специализированного программного обеспечения, что и является основополагающим мотивом создания программного обеспечения различных видов, приведенных на (рис.1):

Виды ПО:

1. Прикладные программы, непосредственно обеспечивающие выполнение необходимых пользователям работ;
2. системные программы, предназначены для управления работой вычислительной системы, выполняют различные вспомогательные функции, например:
3. управление ресурсами компьютера;
4. создание копий используемой информации;
5. проверка работоспособности устройств компьютера;
6. выдача справочной информации о компьютере и др.;
7. инструментальные программные системы, облегчающие процесс создания новых программ для компьютера.

Системное программное обеспечение обеспечивает функционирование и обслуживание компьютера, а также автоматизацию процесса создания новых программ. К системному программному обеспечению относятся: операционные системы и их пользовательский интерфейс; инструментальные программные средства; системы технического обслуживания.

Операционная система — обязательная часть специального программного обеспечения, обеспечивающая эффективное функционирование персонального компьютера в различных режимах, организующая выполнение программ и взаимодействие пользователя и внешних устройств с ЭВМ.

Пользовательский интерфейс (сервисные программы) — это программные надстройки операционной системы (оболочки и среды), предназначенные для упрощения общения пользователя с операционной системой.

Программы, обеспечивающие интерфейс, сохраняют форму общения (диалог) пользователя с операционной системой, но изменяют язык общения (обычно язык команд преобразуется в язык меню). Сервисные системы условно можно разделить на интерфейсные системы, оболочки операционных систем и утилиты.

Интерфейсные системы — это мощные сервисные системы, чаще всего графического типа, совершенствующие не только пользовательский, но и программный интерфейс операционных систем, в частности, реализующие некоторые дополнительные процедуры разделения дополнительных ресурсов.

Оболочки операционных систем предоставляют пользователю качественно новый по сравнению с реализуемым операционной системой интерфейс и делают необязательным знание последнего.

Утилиты автоматизируют выполнение отдельных типовых, часто используемых процедур, реализация которых потребовала бы от пользователя разработки специальных программ. Многие утилиты имеют развитый диалоговый интерфейс с пользователем и приближаются по уровню общения к оболочкам.

Инструментальные программные средства (системы программирования) — обязательная часть программного обеспечения, с использованием которой создаются программы. Инструментальные программные средства включают в свой состав средства написания программ (текстовые редакторы); средства преобразования программ в вид, пригодный для выполнения на компьютере (ассемблеры, компиляторы, интерпретаторы, загрузчики и редакторы связей), средства контроля и отладки программ.

Текстовые редакторы позволяют удобно редактировать, формировать и объединять тексты программ, а некоторые — и контролировать синтаксис создаваемых программ.

Программа, написанная на алгоритмическом языке, должна быть преобразована в объектный модуль, записанный на машинном языке (в двоичных кодах). Подобное преобразование выполняется трансляторами (ассемблером — с языка Assembler и компиляторами — с языков высокого уровня). Для некоторых алгоритмических языков используются интерпретаторы, не создающие объектный модуль, а при каждом очередном выполнении программы, переводящие каждую ее отдельную строку или оператор на машинный язык. Объектный модуль обрабатывается загрузчиком — редактором связей, преобразующие его в исполняемую машинную программу.

Средства отладки позволяют выполнять трассировку программ (пошаговое выполнение с выдачей информации о результатах исполнения), производить проверку синтаксиса программы и промежуточных результатов в точках останова, осуществлять модификацию значений переменных в этих точках.

Системы технического и сервисного обслуживания представляют собой программные средства контроля, диагностики и восстановления работоспособности компьютера, дисков и т. д.

Прикладное программное обеспечение обеспечивает решение пользовательских задач. Ключевым понятием здесь является пакет прикладных программ.

Пакет прикладных программ — это совокупность программ для решения круга задач по определенной тематике или предмету. Различают следующие типы пакетов прикладных программ:

1. общего назначения — ориентированы на автоматизацию широкого круга задач пользователя (текстовые процессоры, табличные редакторы, системы управления базами данных, графические процессоры, издательские системы, системы автоматизации проектирования и т. д.);
2. методо-ориентированные — реализация разнообразных экономико-математических методов решения задач (математического программирования, сетевого планирования и управления, теории массового обслуживания, математической статистики и т. д.);
3. проблемно-ориентированные — направлены на решение определенной задачи (проблемы) в конкретной предметной области (банковские пакеты, пакеты бухгалтерского учета, финансового менеджмента, правовых справочных систем и т. д.).

К прикладному программному обеспечению относятся сервисные программные средства, которые служат для организации удобной рабочей среды пользователя, а также для выполнения вспомогательных функций (информационные менеджеры, переводчики и т. д.).

При построении классификации ПО нужно учитывать тот факт, что стремительное развитие вычислительной техники и расширение сферы приложения компьютеров резко ускорили процесс эволюции программного обеспечения. Если раньше можно было легко перечислить основные категории ПО — операционные системы, трансляторы, пакеты прикладных программ, то сейчас ситуация коренным образом изменилась. Развитие ПО пошло как вглубь (появились новые подходы к построению операционных систем, языков программирования и т.д.), так и вширь (прикладные программы перестали быть прикладными и приобрели самостоятельную ценность). Соотношение между требующимися программными продуктами и имеющимися на рынке меняется очень быстро. Даже классические программные продукты, такие, как операционные системы, непрерывно развиваются и наделяются интеллектуальными функциями, многие из которых ранее относились только к интеллектуальным возможностям человека.

# Инструментальное программное обеспечение

## 2.1 Сущность и понятие инструментального программного обеспечения

Инструментальное программное обеспечение (ИПО) — программное обеспечение, предназначенное для использования в ходе проектирования, разработки и сопровождения программ.

Применяется инструментальное обеспечение в фазе разработки. Инструментальное программное обеспечение — это совокупность программ, используемых для помощи программистам в их работе, для помощи руководителям разработки программного обеспечения в их стремлении проконтролировать процесс разработки и получаемую продукцию. Наиболее известными представителями этой части программного обеспечения являются программы трансляторов с языков программирования, которые помогают программистам писать машинные команды. Инструментальными программами являются трансляторы с языков Фортран, Кобол, Джо-виал, Бейсик, АПЛ и Паскаль. Они облегчают процесс создания новых рабочих программ. Однако трансляторы с языков это только наиболее известная часть инструментальных программ; существует же их великое множество.

Использование вычислительных машин для помощи в создании новых программ далеко не очевидно для людей, не являющихся профессиональными программистами. Часто же бывает так, что профессионалы рассказывают об инструментальном (фаза разработки) и системном (фаза использования) программном обеспечении на едином дыхании, предполагая, что не посвященному в тайны их мастерства известно об этой роли инструментального программного обеспечения. Так же как и в фазе использования (для прикладных программ), системное обеспечение работает и в фазе разработки, но только совместно с инструментальным обеспечением. Инструментальное ПО или системы программирования - это системы для автоматизации разработки новых программ на языке программирования.

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

1. Текстовый редактор для создания файла с исходным текстом программы.

2. Компилятор или интерпретатор. Исходный текст с помощью программы-компилятора переводится в промежуточный объектный код. Исходный текст большой программы состоит из нескольких модулей (файлов с исходными текстами). Каждый модуль компилируется в отдельный файл с объектным кодом, которые затем надо объединить в одно целое.

3. Редактор связей или сборщик, который выполняет связывание объектных модулей и формирует на выходе работоспособное приложение - исполнимый код.

Исполнимый код - это законченная программа, которую можно запустить на любом компьютере, где установлена операционная система, для которой эта программа создавалась. Как правило, итоговый файл имеет расширение .ЕХЕ или .СОМ.

В последнее время получили распространение визуальный методы программирования (с помощью языков описания сценариев), ориентированные на создание Windows-приложений. Этот процесс автоматизирован в средах быстрого проектирования. При этом используются готовые визуальные компоненты, которые настраиваются с помощью специальных редакторов.

Наиболее популярные редакторы (системы программирования программ с использованием визуальных средств) визуального проектирования:

1. Borland Delphi - предназначен для решения практически любых задачи прикладного программирования.
2. Borland C++ Builder - это отличное средство для разработки DOS и Windows приложений.
3. Microsoft Visual Basic - это популярный инструмент для создания Windows-программ.
4. Microsoft Visual C++ - это средство позволяет разрабатывать любые приложения, выполняющиеся в среде ОС типа Microsoft Windows

Таким образом, сущность инструментального программного обеспечения заключается в создании любой исполняемой программы , путем преобразования формально логических выражений в исполняемый машинный код, а также его контроль и корректировка.

## 2.2 Задачи и функции инструментального программного обеспечения

Для инструментального программного обеспечения, как особой разновидности программного обеспечения, характерны общие и частные

функции, как и для всего программного обеспечении в целом. Общие функции рассмотрены нами выше, а специализированными функциями, присущими только данному типу программ, являются:

1. Создание текста разрабатываемой программы с использованием специально установленных кодовых слов (языка программирования), а также определенного набора символов и их расположения в созданном файле - синтаксис программы.

2. Перевод текста создаваемой программы в машинно-ориентированный код, доступный для распознавания ЭВМ. В случае значительного объема создаваемой программы, она разбивается на отдельные модули и каждый из модулей переводится отдельно.

3. Соединение отдельных модулей в единый исполняемый код, с соблюдением необходимой структуры, обеспечение координации взаимодействия отдельных частей между собой.

4. Тестирование и контроль созданной программы, выявление и устранение формальных, логических и синтаксических ошибок, проверка программ на наличие запрещенных кодов, а также оценка работоспособности и потенциала созданной программы.

## 2.3 Виды инструментального программного обеспечения

Исходя из задач, поставленных перед инструментальным программным обеспечением, можно выделить большое количество различных по назначению видов инструментального программного обеспечения:

1. Текстовые редакторы
2. Интегрированные среды разработки
3. SDK
4. Компиляторы
5. Интерпретаторы
6. Линковщики
7. Парсеры и генераторы парсеров (см. Javacc)
8. Ассемблеры
9. Отладчики
10. Профилировщики
11. Генераторы документации
12. Средства анализа покрытия кода
13. Средства непрерывной интеграции
14. Средства автоматизированного тестирования
15. Системы управления версиями и др.

Следует отметить, что оболочки для создания прикладных программ создаются также инструментальными программами и поэтому могут быть отнесены к прикладным программам. Рассмотрим кратко назначения некоторых инструментальных программ.

Текстовые редакторы.

Текстовый редактор — компьютерная программа, предназначенная для обработки текстовых файлов, такой как создание и внесение изменений.

## Типы текстовых редакторов.

Условно выделяют два типа редакторов: потоковые текстовые редакторы и интерактивные.

### Потоковые текстовые редакторы

Потоковые текстовые редакторы представляют собой компьютерные программы, которые предназначены для автоматизированной обработки входных текстовых данных, полученных из текстового файла, в соответствии с заранее заданными пользователями правилами. Чаще всего правила представляют собой регулярные выражения, на специфичном для данного конкретного текстового редактора диалекте. Примером такого текстового редактора может служить редактор Sed.

### Интерактивные текстовые редакторы

Интерактивные текстовые редакторы - это семейство компьютерных программ предназначенных для внесения изменений в текстовый файл в интерактивном режиме. Такие программы позволяют отображать текущее состояние текстовых данных в файле и производить над ними различные действия.

Часто интерактивные текстовые редакторы содержат значительную дополнительную функциональность, призванную автоматизировать часть действий по редактированию, или внести изменение в отображение текстовых данных, в зависимости от их семантики. Примером функциональности последнего рода может служить подсветка синтаксиса.

Текстовые редакторы предназначены для создания и редактирования текстовых документов. Наиболее распространенными являются MS WORD, Лексикон. Основными функциями текстовых редакторов являются:

1. работа с фрагментами документа,
2. вставка объектов созданных в других программах
3. разбивка текста документа на страницы
4. ввод и редактирование таблиц
5. ввод и редактирование формул
6. форматирование абзаца
7. автоматическое создание списков
8. автоматическое создание оглавления.

Известны десятки текстовых редакторов. Наиболее доступными являются NOTEPAD(блокнот), WORDPAD, WORD. Работа конкретного редактора текста определяется обычно функциями, назначение которых отражено в пунктах меню и в справочной системе.

Интегрированная среда разработки

Интегрированная среда разработки, ИСР— система программных средств, используемая программистами для разработки программного обеспечения (ПО). Обычно среда разработки включает в себя:

1. текстовый редактор
2. компилятор и/или интерпретатор
3. средства автоматизации сборки
4. отладчик.

Иногда содержит также средства для интеграции с системами управления версиями и разнообразные инструменты для упрощения конструирования графического интерфейса пользователя. Многие современные среды разработки также включают браузер классов, инспектор объектов и диаграмму иерархии классов — для использования при объектно-ориентированной разработке ПО. Хотя, и существуют среды разработки, предназначенные для нескольких языков программирования — такие, как Eclipse, NetBeans, Embarcadero RAD Studio, Qt Creator или Microsoft Visual Studio, обычно среда разработки предназначается для одного определённого языка программирования - как, например, Visual Basic, Delphi, Dev-C++.

Частный случай ИСР — среды визуальной разработки, которые включают в себя возможность визуального редактирования интерфейса программы.

# SDK.

SDK (от англ. Software Development Kit) или «devkit» — комплект средств разработки, который позволяет специалистам по программному обеспечению создавать приложения для определённого пакета программ, программного обеспечения базовых средств разработки, аппаратной платформы, компьютерной системы, видеоигровых консолей, операционных систем и прочих платформ.

Программист, как правило, получает SDK непосредственно от разработчика целевой технологии или системы. Часто SDK распространяется через Интернет. Многие SDK распространяются бесплатно для того, чтобы поощрить разработчиков использовать данную технологию или платформу.

Поставщики SDK иногда подменяют термин Software в словосочетании Software Development Kit на более точное слово. Например, «Microsoft» и «Apple» предоставляют Driver Development Kits (DDK) для разработки драйверов устройств, а «PalmSource» называет свой инструментарий для разработки «PalmOS Development Kit (PDK)».

## Примеры SDK:

1. Adobe Flex
2. DirectX
3. Eclipse
4. iPhone SDK
5. Java Development Kit
6. Opera Devices SDK
7. Source SDK

Компиляторы.

Компилятор —

1. Программа или техническое средство, выполняющее компиляцию.
2. Машинная программа, используемая для компиляции.
3. Транслятор, выполняющий преобразование программы, составленной на исходном языке, в объектный модуль.
4. Программа, переводящая текст программы на языке высокого уровня в эквивалентную программу на машинном языке.
5. Программа, предназначенная для трансляции высокоуровневого языка в абсолютный код или, иногда, в язык ассемблера. Входной информацией для компилятора (исходный код) является описание алгоритма или программа на проблемно-ориентированном языке, а на выходе компилятора — эквивалентное описание алгоритма на машинно-ориентированном языке (объектный код).

Компиляция —

1. Трансляция программы на язык, близкий к машинному.
2. Трансляция программы, составленной на исходном языке, в объектный модуль. Осуществляется компилятором.

Компилировать — проводить трансляцию машинной программы с проблемно-ориентированного языка на машинно-ориентированный язык.

## Виды компиляторов:

1. Векторизующий. Транслирует исходный код в машинный код компьютеров, оснащённых векторным процессором.
2. Гибкий. Составлен по модульному принципу, управляется таблицами и запрограммирован на языке высокого уровня или реализован с помощью компилятора компиляторов.
3. Диалоговый.
4. Инкрементальный. Повторно транслирует фрагменты программы и дополнения к ней без перекомпиляции всей программы.
5. Интерпретирующий (пошаговый). Последовательно выполняет независимую компиляцию каждого отдельного оператора (команды) исходной программы.
6. Компилятор компиляторов. Транслятор, воспринимающий формальное описание языка программирования и генерирующий компилятор для этого языка.
7. Отладочный. Устраняет отдельные виды синтаксических ошибок.
8. Резидентный. Постоянно находится в основной памяти и доступен для повторного использования многими задачами.
9. Самокомпилируемый. Написан на том же языке, с которого осуществляется трансляция.
10. Универсальный. Основан на формальном описании синтаксиса и семантики входного языка. Составными частями такого компилятора являются: ядро, синтаксический и семантический загрузчики.

## Виды компиляции:

1. Пакетная. Компиляция нескольких исходных модулей в одном пункте задания.
2. Построчная.
3. Условная. Компиляция, при которой транслируемый текст зависит от условий, заданных в исходной программе. Так, в зависимости от значения некоторой константы, можно включать или выключать трансляцию части текста программы.

## Структура компилятора.

Процесс компиляции состоит из следующих этапов:

1. Лексический анализ. На этом этапе последовательность символов исходного файла преобразуется в последовательность лексем.
2. Синтаксический (грамматический) анализ. Последовательность лексем преобразуется в дерево разбора.
3. Семантический анализ. Дерево разбора обрабатывается с целью установления его семантики (смысла) — например, привязка идентификаторов к их декларациям, типам, проверка совместимости, определение типов выражений и т. д. Результат обычно называется «промежуточным представлением/кодом», и может быть дополненным деревом разбора, новым деревом, абстрактным набором команд или чем-то ещё, удобным для дальнейшей обработки.
4. Оптимизация. Выполняется удаление излишних конструкций и упрощение кода с сохранением его смысла. Оптимизация может быть на разных уровнях и этапах — например, над промежуточным кодом или над конечным машинным кодом.
5. Генерация кода. Из промежуточного представления порождается код на целевом языке.

В конкретных реализациях компиляторов эти этапы могут быть разделены или совмещены в том или ином виде.

## Трансляция и компоновка.

Важной исторической особенностью компилятора, отражённой в его названии (англ. compile — собирать вместе, составлять), являлось то, что он мог производить и компоновку (то есть содержал две части — транслятор и компоновщик). Это связано с тем, что раздельная компиляция и компоновка как отдельная стадия сборки выделились значительно позже появления компиляторов. В связи с этим, вместо термина «компилятор» иногда используют термин «транслятор» как его синоним: либо в старой литературе, либо когда хотят подчеркнуть его способность переводить программу в машинный код (и наоборот, используют термин «компилятор» для подчёркивания способности собирать из многих файлов один).

Интерпретаторы.

Интерпретатор (языка программирования) —

1. Программа или техническое средство, выполняющее интерпретацию.
2. Вид транслятора, осуществляющего пооператорную (покомандную) обработку и выполнение исходной программы или запроса (в отличие от компилятора, транслирующего всю программу без её выполнения).
3. Программа (иногда аппаратное средство), анализирующая команды или операторы программы и тут же выполняющая их.
4. Языковый процессор, который построчно анализирует исходную программу и одновременно выполняет предписанные действия, а не формирует на машинном языке скомпилированную программу, которая выполняется впоследствии.

## Типы интерпретаторов.

Простой интерпретатор анализирует и тут же выполняет (собственно интерпретация) программу покомандно (или построчно), по мере поступления её исходного кода на вход интерпретатора. Достоинством такого подхода является мгновенная реакция. Недостаток — такой интерпретатор обнаруживает ошибки в тексте программы только при попытке выполнения команды (или строки) с ошибкой.

Интерпретатор компилирующего типа — это система из компилятора, переводящего исходный код программы в промежуточное представление, например, в байт-код или p-код, и собственно интерпретатора, который выполняет полученный промежуточный код (так называемая виртуальная машина). Достоинством таких систем является большее быстродействие выполнения программ (за счёт выноса анализа исходного кода в отдельный, разовый проход, и минимизации этого анализа в интерпретаторе). Недостатки — большее требование к ресурсам и требование на корректность исходного кода. Применяется в таких языках, как Java, PHP, Python, Perl (используется байт-код), REXX (сохраняется результат парсинга исходного кода), а также в различных СУБД (используется p-код).

В случае разделения интерпретатора компилирующего типа на компоненты получаются компилятор языка и простой интерпретатор с минимизированным анализом исходного кода. Причём исходный код для такого интерпретатора не обязательно должен иметь текстовый формат или быть байт-кодом, который понимает только данный интерпретатор, это может быть машинный код какой-то существующей аппаратной платформы. К примеру, виртуальные машины вроде QEMU, Bochs, VMware включают в себя интерпретаторы машинного кода процессоров семейства x86.

Некоторые интерпретаторы (например, для языков Лисп, Scheme, Python, Бейсик и других) могут работать в режиме диалога или так называемого цикла чтения-вычисления-печати (англ. read-eval-print loop, REPL). В таком режиме интерпретатор считывает законченную конструкцию языка (например, s-expression в языке Лисп), выполняет её, печатает результаты, после чего переходит к ожиданию ввода пользователем следующей конструкции.

Уникальным является язык Forth, который способен работать как в режиме интерпретации, так и компиляции входных данных, позволяя переключаться между этими режимами в произвольный момент, как во время трансляции исходного кода, так и во время работы программ.

Следует также отметить, что режимы интерпретации можно найти не только в программном, но и аппаратном обеспечении. Так, многие микропроцессоры интерпретируют машинный код с помощью встроенных микропрограмм, а процессоры семейства x86, начиная с Pentium (например, на архитектуре Intel P6), во время исполнения машинного кода предварительно транслируют его во внутренний формат (в последовательность микроопераций).

## Алгоритм работы простого интерпретатора:

1. прочитать инструкцию;
2. проанализировать инструкцию и определить соответствующие действия;
3. выполнить соответствующие действия;
4. если не достигнуто условие завершения программы, прочитать следующую инструкцию и перейти к пункту 2.

## Достоинства и недостатки интерпретаторов.

### Достоинства:

1. Большая переносимость интерпретируемых программ — программа будет работать на любой платформе, на которой есть соответствующий интерпретатор.
2. Как правило, более совершенные и наглядные средства диагностики ошибок в исходных кодах.
3. Упрощение отладки исходных кодов программ.
4. Меньшие размеры кода по сравнению с машинным кодом, полученным после обычных компиляторов.

### Недостатки:

1. Интерпретируемая программа не может выполняться отдельно без программы-интерпретатора. Сам интерпретатор при этом может быть очень компактным.
2. Интерпретируемая программа выполняется медленнее, поскольку промежуточный анализ исходного кода и планирование его выполнения требуют дополнительного времени в сравнении с непосредственным исполнением машинного кода, в который мог бы быть скомпилирован исходный код.
3. Практически отсутствует оптимизация кода, что приводит к дополнительным потерям в скорости работы интерпретируемых программ.

# Компоновщик.

Компоновщик (также редактор связей, линкер) — программа, которая производит компоновку — принимает на вход один или несколько объектных модулей и собирает по ним исполнимый модуль.

Для связывания модулей компоновщик использует таблицы имён, созданные компилятором в каждом из объектных модулей. Такие имена могут быть двух типов:

1. Определённые или экспортируемые имена — функции и переменные, определённые в данном модуле и предоставляемые для использования другим модулям.
2. Неопределённые или импортируемые имена — функции и переменные, на которые ссылается модуль, но не определяет их внутри себя.

Работа компоновщика заключается в том, чтобы в каждом модуле разрешить ссылки на неопределённые имена. Для каждого импортируемого имени находится его определение в других модулях, упоминание имени заменяется на его адрес.

Компоновщик обычно не выполняет проверку типов и количества параметров процедур и функций. Если надо объединить объектные модули программ, написанные на языках со строгой типизацией, то необходимые проверки должны быть выполнены дополнительной утилитой перед запуском редактора связей.

# Ассемблер.

Ассемблер (от англ. assembler — сборщик) — компьютерная программа, компилятор исходного текста программы, написанной на языке ассемблера, в программу на машинном языке.

Как и сам язык (ассемблера), ассемблеры, как правило, специфичны конкретной архитектуре, операционной системе и варианту синтаксиса языка. Вместе с тем существуют мультиплатформенные или вовсе универсальные (точнее, ограниченно-универсальные, потому что на языке низкого уровня нельзя написать аппаратно-независимые программы) ассемблеры, которые могут работать на разных платформах и операционных системах. Среди последних можно также выделить группу кросс-ассемблеров, способных собирать машинный код и исполняемые модули (файлы) для других архитектур и ОС.

Ассемблирование может быть не первым и не последним этапом на пути получения исполняемого модуля программы. Так, многие компиляторы с языков программирования высокого уровня выдают результат в виде программы на языке ассемблера, которую в дальнейшем обрабатывает ассемблер. Также результатом ассемблирования может быть не исполняемый, а объектный модуль, содержащий разрозненные и непривязанные друг к другу части машинного кода и данных программы, из которого (или из нескольких объектных модулей) в дальнейшем с помощью программы-компоновщика («линкера») может быть скомпонован исполнимый файл.

Отладчик или дебаггер является модулем среды разработки или отдельным приложением, предназначенным для поиска ошибок в программе. Отладчик позволяет выполнять пошаговую трассировку, отслеживать, устанавливать или изменять значения переменных в процессе выполнения программы, устанавливать и удалять контрольные точки или условия остановки и т. д.

## Список отладчиков.

1. AQtime — коммерческий отладчик для приложений, созданных для .NET Framework версии 1.0, 1.1, 2.0, 3.0, 3.5 (включая ASP.NET приложения), а также для Windows 32- и 64-битных приложений.
2. DTrace — фреймворк динамической трассировки для Solaris, OpenSolaris, FreeBSD, Mac OS X и QNX.
3. Electric Fence — отладчик памяти.
4. GNU Debugger (GDB) — отладчик программ от проекта GNU.
5. IDA — мощный дизассемблер и низкоуровневый отладчик для операционных систем семейства Windows и Linux.
6. Microsoft Visual Studio — среда разработки программного обеспечения, включающая средства отладки от корпорации Microsoft.
7. OllyDbg — бесплатный низкоуровневый отладчик для операционных систем семейства Windows.
8. SoftICE — низкоуровневый отладчик для операционных систем семейства Windows.
9. Sun Studio — среда разработки программного обеспечения, включающая отладчик dbx для ОС Solaris и Linux, от корпорации Sun Microsystems.
10. Dr. Watson — стандартный отладчик Windows, позволяет создавать дампы памяти.
11. TotalView — один из коммерческих отладчиков для UNIX.
12. WinDbg — бесплатный отладчик от корпорации Microsoft.

Генератор документации — программа или пакет программ, позволяющая получать документацию, предназначенную для программистов (документация на API) и/или для конечных пользователей системы, по особым образом комментированному исходному коду и, в некоторых случаях, по исполняемым модулям (полученным на выходе компилятора).

Обычно генератор анализирует исходный код программы, выделяя синтаксические конструкции, соответствующие значимым объектам программы (типам, классам и их членам/свойствам/методам, процедурам/функциям и т. п.). В ходе анализа также используется мета-информация об объектах программы, представленная в виде документирующих комментариев. На основе всей собранной информации формируется готовая документация, как правило, в одном из общепринятых форматов — HTML, HTMLHelp, PDF, RTF и других.

## Документирующие комментарии.

Документирующий комментарий — это особым образом оформленный комментарий к объекту программы, предназначенный для использования каким-либо конкретным генератором документации. От того, какой генератор документации применяется, зависит синтаксис конструкций, используемых в документирующих комментариях.

В документирующих комментариях может содержаться информация об авторе кода, описываться назначение объекта программы, смысл входных и выходных параметров — для функции/процедуры, примеры использования, возможные исключительные ситуации, особенности реализации.

Документирующие комментарии, как правило, оформляются как многострочные комментарии в стиле языка Си. В каждом случае комментарий должен находиться перед документируемым элементом. Первым символом в комментарии (и вначале строк комментария) должен быть \*. Блоки разделяются пустыми строками.

# Visual Basic for Applications

программный обеспечение операционный системный

## 3.1 Сущность Visual Basic и его краткая история

Microsoft Visual Basic (VB) — средство разработки программного обеспечения, разрабатываемое корпорацией Microsoft и включающее язык программирования и среду разработки. Язык Visual Basic унаследовал дух, стиль и отчасти синтаксис своего предка — языка Бейсик, у которого есть немало диалектов. В то же время Visual Basic сочетает в себе процедуры и элементы объектно-ориентированных и компонентно-ориентированных языков программирования. Среда разработки VB включает инструменты для визуального конструирования пользовательского интерфейса. (см. табл.).

Visual Basic (основные характеристики)

|  |  |
| --- | --- |
| Класс языка: | процедурный, объектно-ориентированный, компонентно-ориентированный |
| Тип исполнения: | компилируемый, интерпретируемый |
| Появился в | 1991 г. |
| Релиз: | Visual Basic 6.0 (1998) |
| Типизация данных: | не строгая |
| Основные реализации: | Microsoft Visual Basic for DOS, Microsoft Visual Basic for Windows |
| Диалекты: | Visual Basic for Application, 1993; Visual Basic Script, 1996 |
| Испытал влияние: | QuickBasic, 1985 |
| Повлиял на: | VB.net, 2001 |

Visual Basic считается хорошим средством быстрой разработки прототипов программы, для разработки приложений баз данных и вообще для компонентного способа создания программ, работающих под управлением операционных систем семейства Microsoft Windows.

## В процессе эволюции Visual Basic прошел ряд последовательных этапов, позволивших ему стать одним из самых популярных языков программирования на сегодняшний день. Итак, эволюция Visual Basic шла следующим путем:

1. май 1991 — выпущен Visual Basic 1.0 для Microsoft Windows. За основу языка был взят синтаксис QBasic, а новшеством, принесшим затем языку огромную популярность, явился принцип связи языка и графического интерфейса.
2. сентябрь 1992 — выпущен Visual Basic 1.0 под DOS. Он не был полностью совместим с Windows-версией VB, поскольку являлся следующей версией QuickBASIC и работал в текстовом режиме экрана.
3. ноябрь 1992 — выпущен Visual Basic 2.0. Среда разработки стала проще в использовании и работала быстрее.
4. летом 1993 — вышел в свет Visual Basic 3.0 в версиях Standard и Professional. Ко всему прочему, в состав поставки добавился движок для работы с базами данных Access.
5. август 1995 — Visual Basic 4.0 — версия, которая могла создавать как 32-х, так и 16-разрядные Windows-программы.
6. февраль 1997 — Visual Basic 5.0 — начиная с этой версии, стало возможно, наряду с обычными приложениями, разрабатывать COM-компоненты.
7. В середине 1998 — вышла Visual Basic 6.0. После этого Microsoft резко изменила политику в отношении языков семейства Basic. Вместо развития Visual Basic, был создан абсолютно новый язык Visual Basic .NET.
8. В 2005 году вышла новая версия Visual Basic, в комплекте Visual Studio. Порадовала она новым интерфейсом и возможностями. Язык основан на Visual Basic.NET.
9. В конце 2007 Microsoft выпустила новую версию Visual Basic — Visual Basic 2008, которая также была основана на Visual Basic.NET.

Исходя из функциональных возможностей и специфики применения, можно выделить следующие разновидности указанной программы:

* 1. Классический Visual Basic (версии 5-6) Этот язык очень сильно привязан к своей среде разработки и к операционной системе Windows, являясь исключительно инструментом написания Windows-приложений
	2. Visual Basic for Applications (VBA) Это средство программирования, практически ничем не отличающееся от классического Visual Basic, которое предназначено для написания макросов и других прикладных программ для конкретных приложений. Наибольшую популярность получил благодаря своему использованию в пакете Microsoft Office. Широкое распространение Visual Basic for Applications в сочетании с изначально недостаточным вниманием к вопросам безопасности привело к широкому распространению макровирусов.
	3. Visual Basic Scripting Edition (VBScript) Скриптовый язык, являющийся несколько усечённой версией обычного Visual Basic. Используется в основном для автоматизации администрирования систем Windows, а также для создания страниц ASP и сценариев для Internet Explorer.

## 3.2 Visual Basic for Application интерфейс, основные функции и возможности

Создавая Visual Basic for Application, корпорация Microsoft ставила своей основной задачей создание инструментального обеспечения, доступного для пользователей, не являющихся профессиональными программистами, но в то же время достаточно квалифицированных для разработки и проектирования прикладных программ и приложений на базе Microsoft Office. Именно решая указанную задачу, разработчики создали VBA, наделив его рядом уникальных особенностей. Одной из таких, наиболее ценных для пользователя является возможность создавать и использовать в программах нестандартные (настраиваемые) диалоговые окна, добавляя объект UserForm в проект, а так же удобный пользовательский интерфейс.

Интерфейс программы Visual Basic for Application состоит из комплекса различных окон и вкладок, используемых при проектировании создаваемого приложения, основными из которых являются:

1. окно Проекта (рис.2), отображающее структуру создаваемого проекта.
2. окно Программного кода (рис. 3), отображающее программный код создаваемого проекта и дающее возможность писать программу классическим способом при помощи встроенного редактора кодовых слов, которых в VBA более 16 тысяч. Также данное окно позволяет редактировать код и проверять его на наличие ошибок.
3. закладка Свойств (рис. 4), отображающая установленные к указанному объекту параметры и дающая возможность изменить указанные настройки.

Перемещаясь между окнами и закладками, пользователь может легко настраивать созданный проект.

Используя создаваемые пользователем формы VBA, можно создавать нестандартные диалоговые окна для отображения данных или получения значений от пользователя программы в том виде, который наиболее соответствует потребностям программы. Например, можно создать тест, отобразить диалоговое окно для отображения вопросов с вариантами ответов и предоставить пользователю возможность выбрать один из вариантов ответа, который он считает верным.

Нестандартные диалоговые окна позволяют программе взаимодействовать с её пользователем самым сложным образом и обеспечивают разнообразную форму ввода и вывода данных.

Нестандартное диалоговое окно создаётся в VBA посредством добавления объекта UserForm в проект. Этот объект представляет собой пустое диалоговое окно; оно имеет строку заголовка и кнопку закрытия, но в нём отсутствуют какие-либо другие элементы управления. Нестандартное диалоговое окно создаётся путем добавления элементов управления в объект UserForm и обычно называемый просто формой (Рис. 5).

Каждый объект UserForm имеет свойства, методы и события, наследуемые им от класса объектов UserForm.

Каждый объект UserForm также содержит модуль класса, в который пользователь добавляет собственные методы и свойства или вписывает процедуры обработки событий для данной формы.

Возможность создавать создать собственный интерфейс, независимый от среды программы-приложения, например Excel, при помощи экранных форм является одной из наиболее ценных возможностей в VBA.

Экранные формы - это окна различного назначения и вида, созданные пользователем для своего приложения. Они содержат элементы управления, позволяющие пользователю обмениваться информацией с приложением.

VBA использует созданный графический дизайн формы - с настройками свойств формы и элементов управления - для получения всей информации, необходимой для отображения диалогового окна: размеров диалогового окна, элементов управления в нём и т.п. В результате VBA позволяет отобразить форму диалогового окна с помощью единственной инструкции.

Для отображения нестандартного диалогового окна используется метод Show объекта UserForm. Если в настоящий момент форма не загружена в память, метод Show загружает форму и отображает её. Если форма уже загружена, метод Show просто отображает её.

Отображения одного диалогового окна для выполнения задачи обычно недостаточно. Почти всегда требуется определить состояние элементов управления диалогового окна с целью выяснить, какие данные или опции выбрал пользователь. Например, если диалоговое окно используется для получения от пользователя информации о том, по каким столбцам и строкам должно выполняться упорядочение рабочего листа, необходимо иметь возможность выяснить, какие значения пользователь ввел после закрытия диалогового окна и до действительного начала операции упорядочивания.

В других случаях может потребоваться динамическое изменение заголовков кнопок (или других элементов управления) диалогового окна, динамическое обновление надписи или поля, связанного со счетчиком, или динамическое подтверждение введенных в диалоговое окно данных.

В VBA появляется возможность значительно расширить набор функций встроенных в стандартное приложение, например Microsoft Excel, а также создавать функции, значения которых зависят от некоторых условий и событий.

VBA позволяет программировать табличные функции. Чтобы создать отдельный рабочий лист для программного модуля, предусмотрена закладка Insert Module из меню Visual, команда Module из меню Insert Macro. После этого появится новый рабочий лист "Modele1". В программном модуле нужно описать функцию на языке VBA. В окне программного модуля можно работать, как в окне небольшого текстового редактора.

Встраивание функций осуществляется командой Object Browser из меню View. Функции, определенные пользователем, рассматриваются в программе как самостоятельные объекты. VBA обладает значительным набором встроенных функций, разделяя их на типы.

Visual Basic позволяет резервировать переменные, с указанием размера и без него, работать с различными типами данных, использовать константы, работать с математическими операторами и функциями, использовать дополнительные операторы. Предусмотрено использование операторов циклов For Next, Do, объектов типа “таймер” (невидимый секундомер в программе). Точность установления времени в программе составляет 1 миллисекунду, или 1/1000 сек. Запущенный таймер постоянно работает - т.е. выполняется соответствующая процедура обработки прерывания через заданный интервал времени - до тех пор, пока пользователь не остановит таймер или не отключит программу.

В VBA можно задать любое свойство для формы, включая заголовок, размер, тип рамки, цвет фона и символов, шрифт текста и фоновый рисунок.

Если обобщить все функции программы, то Visual Basic for Application позволяет:

1. работать со средствами управления
2. работать с меню и диалоговыми окнами
3. работать с формами, принтерами и обработчиками ошибок
4. добавлять художественное оформление и спецэффекты
5. использовать модули и процедуры
6. работать с наборами элементов и массивами
7. работать с текстовыми файлами и базами данных
8. взаимодействовать с Microsoft Office.

Помимо этого Visual Basic for Application предоставляет три очень полезных объекта для предоставления доступа к файловой системе. Это списки дисковых накопителей, которые позволяют пролистывать доступные диски в системе; списки директорий, которые позволяют ориентироваться в папках на выбранном диске, и списки файлов, позволяющие выбрать нужный файл в папке.

## 3.3 Достоинства, недостатки и критика Visual Basic for Application

### Достоинства:

1. Высокая скорость создания приложений с графическим интерфейсом для MS Windows.
2. Простой синтаксис, позволяющий очень быстро освоить язык.
3. Возможность компиляции как в машинный код, так и в P-код (по выбору программиста). В режиме отладки программа всегда (вне зависимости от выбора) компилируется в P-код, что позволяет приостанавливать выполнение программы, вносить значительные изменения в исходный код, а затем продолжать выполнение: полная перекомпиляция и перезапуск программы при этом не требуется.
4. Защита от ошибок, связанных с применением указателей и доступом к памяти. Этот аспект делает Visual Basic приложения более стабильными, но также является объектом критики.
5. Возможность использования большинства WinAPI функций для расширения функциональных возможностей приложения. Данный вопрос наиболее полно исследован Дэном Эпплманом, написавшим книгу «Visual Basic Programmer's Guide to the Win32 API».

### Недостатки:

1. Поддержка операционных систем только семейства Windows и Mac OS X (Исключение — VB1 for DOS).
2. Отсутствие механизма наследования реализации объектов[2]. Существующее в языке наследование позволяет наследовать только интерфейсы, но не их реализацию.
3. Требует установленную msvbvmXX.dll для работы программы.
4. Медленная скорость работы, обусловленная тем, что практически все встроенные функции языка реализованы через библиотеку времени исполнения (runtime library), которая, в свою очередь, производит много «лишней» работы по проверке и/или преобразованию типов.

### Критика:

1. Часто критике подвергаются такие аспекты Visual Basic, как возможность отключить средства слежения за объявленными переменными, возможность неявного преобразования переменных, наличие типа данных «Variant». По мнению критиков, это даёт возможность писать крайне плохой код. С другой стороны, это можно рассматривать как плюс, так как VB не навязывает «хороший стиль», а даёт больше свободы программисту.
2. Отсутствие указателей, низкоуровневого доступа к памяти, ASM-вставок. Несмотря на то, что парадигма Visual Basic позволяет среднему VB-программисту обходиться без всего этого, перечисленные вещи также нередко становятся объектами критики. И хотя, используя недокументированные возможности и определённые ухищрения, всё это можно реализовать и на VB (например, с помощью функций для получения указателей VarPtr(), StrPtr() и ObjPtr()); пользоваться этими трюками гораздо сложнее, чем, например, на Си++.

Однако стоит отметить, что все недостатки языка вытекают из его основного достоинства — простоты разработки графического интерфейса. Поэтому многие программисты используют Visual Basic для разработки интерфейса пользователя, а функциональность программы реализуют в виде динамически подключаемых библиотек (DLL), написанных на другом языке (чаще всего C++).

# Практическая часть

##

## 4.1 Постановка задачи

Составить блок-схему и написать программу на языке Pascal. Рассчитать внутреннюю стоимость ценных бумаг. Внутренняя стоимость актива определяется будущим потоком доходов от этого актива

pv – текущая внутренняя стоимость акции

c – ожидаемое поступление от рассматриваемого актива

r – норма доходности, ожидаемая инвестором для дохода с соответствующим уровнем риска

n – фактор времени (в месяцах).

Выполнить анализ рынка и упорядочить результат по возрастанию полученных данных.

##

## 4.2 Текст программы на языке Pascal

Program Ann;

var

pv: array [1. .5] of real;

x, s: real;

i, j: integer;

c, r: real;

begin

s:=0

for i:=1 to 5 do

begin

writeLn (‘Введите ожидаемое поступление от ‘,i,’-го актива c:’);

readLn (c);

writeLn (‘Введите норму доходности, ожидаемую инвестором r:’);

readLn (r);

pv:=c/exp(ln(1+r)\*i);

writeLn (‘текущая внутренняя стоимость актива равна’, pv[i]:1:3);

s:=s+pv[i];

end;

writeLn (‘Внутренняя стоимость актива равна’, s);

for i=1 to 5 do

begin

for j:=1 to 4 do

begin

if pv[j] > pv[j+1] then

begin

x:=pv[j];

pv[j]:=pv[j+1];

pv[j+1]:=x;

end;

end;

end;

readLn;

writeLn (‘Стоимость активов, отсортированная по возрастанию’);

for i:=1 to 5 do

begin

writeLn (pv[i]:1:3);

end;

readLn;

end.

## Контрольный пример


## Результат выполнения программы на контрольном примере


# Заключение

Итак, подытожив все вышесказанное, следует отметить, что инструментальное программное обеспечение является одним из видов программного обеспечения, обладая его общими задачами и функциями.

Однако, являясь узкоспециализированным видом программного обеспечения, обладает определенным набором уникальных свойств и функций, обеспечивающих решение свойственных ему задач.

Необходимо отметить наметившуюся тенденцию к упрощению процесса программирования и создания определенного подкласса – полупрофессиональное программирование для прикладных целей.

Именно это позволит опытному пользователю компьютера, но не профессиональному программисту, создавать некие приложения и небольшие исполняемые в среде Microsoft Office файлы, используемые в первую очередь для целей учета и обеспечения документооборота в небольших компаниях.

Именно с этой целью Microsoft был разработан программный комплекс Visual Basic for Application, позволяющий облегчить процесс программирования и давший возможность заниматься прикладным программированием пользователям, а не программистам. Данная возможность была реализована в первую очередь путем создания раздела программы – «Редактор сценариев» и возможности записывать и исполнять «Макросы», как отдельную разновидность графически программируемых модулей. Реализована возможность создания приложений с графическим интерфейсом для MS Windows. Также достоинством данного вида инструментального программного обеспечения является простой синтаксис, позволяющий очень быстро освоить язык, и применять его для программирования во всех стандартных приложениях Microsoft Office.

Поэтому трудно переоценить значение инструментального обеспечения в целом, и Visual Basic for Application в частности, хотя недостатки, а о них было сказано выше, также имеют место. Но это скорее даже не негативные стороны продукта, а ориентиры для дальнейшего совершенствования инструментального обеспечения в лице Visual Basic for Application.

# Список использованных источников

1. Алгоритмические языки реального времени /Под ред. Янга С./ 2004 г.
2. Журнал PC Magazine Russian Edition №2 2008г. Компьютер сегодня.
3. Информатика. /Под ред. Могилев А.В., Пак Н.И., Хеннер Е.К/ – М.: ACADEMIA, 2000.
4. Информатика и информационные технологии: Учебник /Под ред. Романова Д.Ю./ ООО «издательство «Эксмо», 2007.
5. Новейшая энциклопедия персонального компьютера /Под ред. Леонтьева В. /Москва, 1999 год. – 271 с.
6. Новые языки программирования и тенденции их развития /Под ред. Ушковой В./ 2001 г.
7. Педагогика /Под ред. Пидкасистого П.И./ – М.: Педагогическое общество Россия, 2000.
8. Программирование для Microsoft Excel 2000 за 21 день. /Под ред. Хариса М./ – М.: Вильямс, 2000.
9. Симонович С. Информатика: базовый курс. Учеб. для ВУЗов. СПб, Питер, 2002 г.
10. С Excel 2000 без проблем. /Под ред. Ковальски/ – М.: Бином, 2000.
11. «Эффективная работа в Windows 98» /Под ред. Стинсона К./ 2000 год. – 247 с.
12. Языки программирования. кн.5 /Под ред. Ваулина А.С./ 2003 г.
13. Языки программирования: разработка и реализация /Под ред. Терренса П./ 2001 г.
14. Электронный учебник по информатике. Алексеев Е.Г. http://www.stf.mrsu.ru/economic/lib/Informatics/text/Progr.html\
15. http://www.wikipedia.org