Министерство науки и высшего образования РФ
Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

КУРСОВОЙ ПРОЕКТ

по дисциплине «Цифровой практикум»

СИСТЕМЫ ПРОГРАММИРОВАНИЯ

Выполнил
студент гр. 5130904/30005 Кладковой М.Д.

Руководитель
доцент, к.т.н. Тутыгин В. С.

Санкт-Петербург
2023

Введение

Оглавление

Введение

В пятидесятые годы двадцатого века с появлением компьютеров на электронных лампах началось бурное развитие систем программирования. К сегодняшнему дню насчитывают несколько поколений систем программирования. Каждое из последующих поколений по своей функциональной мощности качественно отличается от предыдущего. С появлением персональных компьютеров системы стали составными частями интегрированных сред разработки. Появились системы, применяемые в различных офисных программах. В настоящее время системы программирования применяются в самых различных областях человеческой деятельности, таких как научные вычисления, системное программирование, обработка информации, искусственный интеллект, издательская деятельность, удаленная обработка информации, описание документов.

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

Целью данной работы есть изучение существующих систем программирования и в результате изучения и понимание процесса их использования. Основными задачами данной работы является:

1. Анализ литературы по избранной теме
2. Определение системы программирования
3. Анализ классификации и состава систем программирования
4. Описание основных существующих систем программирования
5. Вывод на основе анализируемых данных

Определение системы программирования

Достаточно важная часть любого компьютера – система программирования. Что это такое? Система программирования — это базовый комплекс программного обеспечения, поддерживающий процесс программирования. Системы программирования представляют собой единство инструментальных и исполнительных средств. Данная система обычно включает в себя следующие компоненты:

- Файловая система для хранения текста программ - как правило, это общая часть программного обеспечения для различных систем на данном компьютере.
- Редактор для ввода текста программы как последовательности символов и исправление её (текстовый редактор). При этом возможно как использование редактора, специализированного для составления программ на данном языке, так и универсального, предназначенного для набора различных текстов.
- Транслятор для преобразования текста программы к виду, в котором она может исполняться, и указания ошибок, если преобразование не удаётся. Транслятором может быть не одна программа.
- Библиотеки периода трансляции, которые используются в процессе преобразования программного текста, к примеру, для включения в него стандартизованных фрагментов (чтобы программисту не нужно было их повторять в своих программных текстах).
- Библиотеки периода исполнения, содержащие программы стандартных действий абстрактного вычислителя (её еще называют библиотека поддержки языка). Они связывают язык с операционной средой.
- Отладчик - программа, отслеживающая ход вычислений программ на данном языке. С его помощью можно последовательно выполнять отдельные операторы исходного

текста по шагам, наблюдая при этом, как меняются значения различных переменных. Без отладчика разработать крупное приложение очень сложно.

- Пользовательские библиотеки, которые содержат программы на данном языке (в текстовом или преобразованном виде), используемые в составляемых программах для задания специальных вычислений (они зависят от среды программирования).
- Редакторы внешних связей, собирающие программы из модулей.
- Загрузчики
- Оптимизаторы, позволяющие автоматически улучшать программу, написанную на определённом языке
- Профилировщики, которые определяют, какой процент времени выполняется та или иная часть программы. Это позволяет выявить наиболее интенсивно используемые фрагменты программы и оптимизировать их (например, переписав на языке Ассемблера)
- Язык или языки программирования, на котором будет создаваться программный код

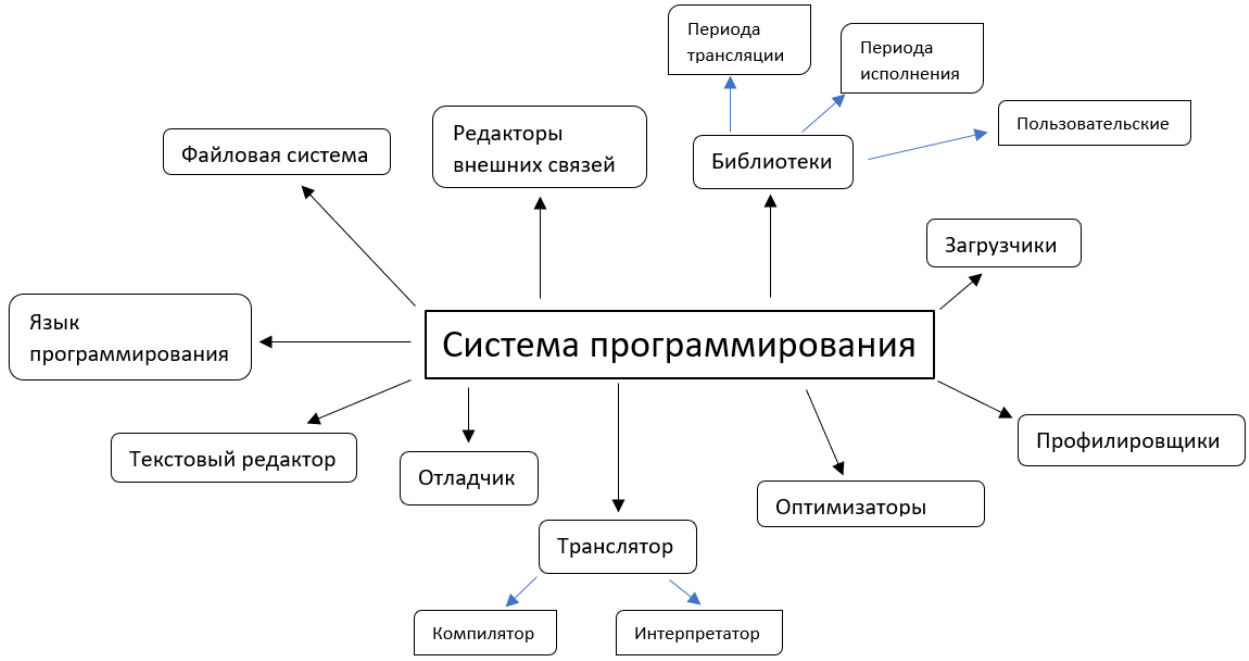

Рис. 1 Система программирования

На сегодняшний день есть представление о следующем традиционном составе системы программирования. В любой из сфер жизнедеятельности человека: медицина, образование, наука, проектирование зданий, пищевая промышленность, торговля – используются компьютерные технологии. В любой из сфер, если есть знание хоть одного языка программирования, это значительно упростит вашу работу, развитие и рост результата. Программное обеспечение различного предназначения написано на языках программирования. Также очень важно отметить, что на сегодняшний день программное обеспечение является сложным продуктом и реализуется на нескольких языках программирования одновременно. Изучение программирования в свою очередь влечет необходимость изучения или ознакомления со смежными дисциплинами и науками: теория алгоритмов, компьютерная графика, системный анализ.

Для чего нужны системы программирования? Они позволяют программистам заниматься разработкой компьютерных программ. Системы программирования — это универсальные средства работы с информацией. С их помощью можно решать вычислительные задачи, обрабатывать тексты, получать графические изображения, осуществлять хранение и поиск данных и т. д., в общем, делать все, что делают средства прикладного программного обеспечения - специализированные исполнители. Кроме того, сами эти средства (графические и текстовые

редакторы, СУБД[1] и др.) — это программы, написанные на языках программирования, созданные с помощью систем программирования.

При изучении систем программирования следует также различать следующие понятия:

1. Программный инструмент – программа, которая используется для поддержки разработки программных продуктов. Например, отладчик, который позволяет программисту проводить тестирование и исправление ошибок.
2. Утилита – программа, которая предоставляет общие функции (копирование файлов, подготовка текстов, организацию ссылок и т. д.).
3. Библиотеки процедур – набор процедур различного назначения, которые существенно упрощают разработку приложений. Например, библиотека функций ввода-вывода или математических функций.

Классификация систем программирования

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

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

Машинно-ориентированные системы

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

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

Классификация машинно-ориентированных систем программирования:

1. Машинные языки — совокупность машинных команд, отличающаяся количеством адресов в команде, назначением информации, задаваемой в адресах, набором операций, которые может выполнять машина. Каждый компьютер имеет свой машинный язык
2. Языки символического кодирования — они схожи с машинными языками и являются командными, однако представляют собой не последовательности двоичных и восьмеричных цифр, а символический код в виде идентификаторов, предназначенные для облегчения запоминания смыслового содержания операции.
3. Автокод — языки, включающие в себя все возможности символического кодирования, посредством расширенного введения макрокоманд. Макрокоманда — программный алгоритм действий, записанный пользователем.

[1] Система управления базами данных (сокр. СУБД) - комплекс программ, позволяющих создать базу данных (БД) и манипулировать данными (вставлять, обновлять, удалять и выбирать). Система обеспечивает безопасность, надёжность хранения и целостность данных, а также предоставляет средства для администрирования БД.

4. Макрос — набор команд и инструкций, группируемых вместе в виде единой команды для автоматического выполнения задачи. Основное назначение макроса — сокращение последовательности символов, описывающих выполнение требуемых действий ЭВМ, для более сжатого вида.

Машинно-независимые системы программирования

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

Классификация систем программирования

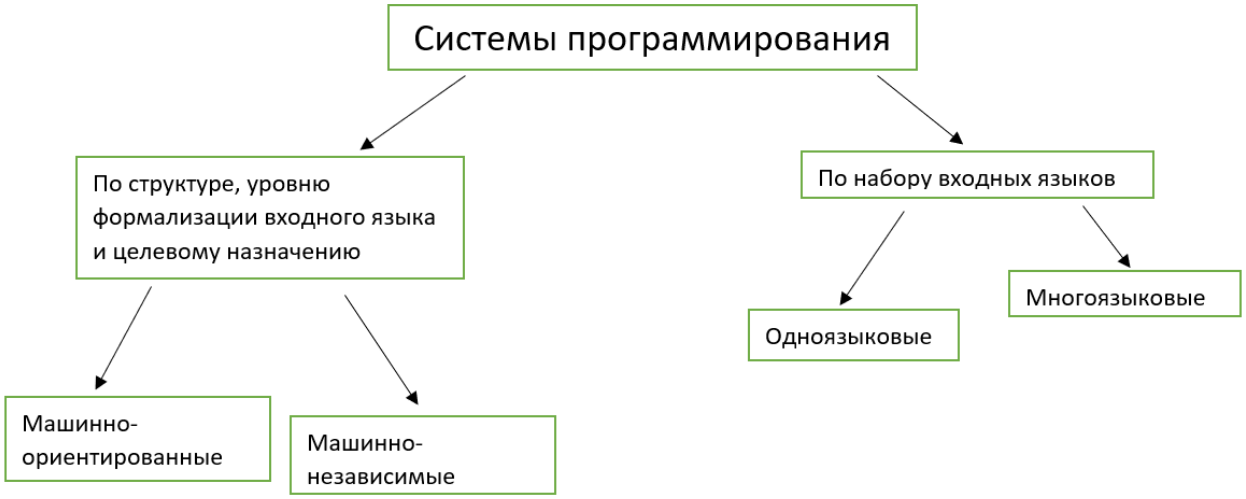

Рис. 2 Классификация систем программирования

Состав систем программирования

Язык программирования

Язык программирования - формальный язык, предназначенный для записи компьютерных программ. Язык программирования определяет набор лексических, синтаксических и семантических правил, определяющих внешний вид программы и действия, которые выполнит исполнитель (обычно — ЭВМ) под её управлением.

Со времени создания первых программируемых машин человечество придумало более восьми тысяч языков программирования (включая эзотерические, визуальные и игрушечные). Каждый год их число увеличивается. Некоторыми языками умеет пользоваться только небольшое число их собственных разработчиков, другие становятся известны миллионам людей. Профессиональные программисты могут владеть несколькими языками программирования.

Состав систем программирования

Язык программирования предназначен для написания компьютерных программ, которые представляют собой набор правил, позволяющих компьютеру выполнить тот или иной вычислительный процесс, организовать управление различными объектами, и т. п. Язык программирования отличается от естественных языков тем, что предназначен для управления ЭВМ, в то время как естественные языки используются, прежде всего, для общения людей между собой. Большинство языков программирования использует специальные конструкции для определения и манипулирования структурами данных и управления процессом вычислений.

Как правило, язык программирования определяется не только через спецификации стандарта языка, формально определяющие его синтаксис и семантику, но и через воплощения (реализации) стандарта — программные средства, обеспечивающие трансляцию или интерпретацию программ на этом языке; такие программные средства различаются по производителю, марке и варианту (версии), времени выпуска, полноте воплощения стандарта, дополнительным возможностям; могут иметь определённые ошибки или особенности воплощения, влияющие на практику использования языка или даже на его стандарт.

Классификация языков программирования

Существует множество признаков, по которым можно разделить языки программирования:

1. Компилируемые и интерпретируемые.

Программа на компилируемом языке при помощи специальной программы компилятора преобразуется (компилируется) в набор инструкций для данного типа процессора (машинный код) и далее записывается в исполняемый файл, который может быть запущен на выполнение как отдельная программа. Другими словами, компилятор переводит программу с языка высокого уровня на низкоуровневый язык, понятный процессору сразу и целиком, создавая при этом отдельную программу

Как правило, скомпилированные программы выполняются быстрее и не требуют для выполнения дополнительных программ, так как уже переведены на машинный язык. Вместе с тем при каждом изменении текста программы требуется ее перекомпиляция, что создает трудности при разработке. Кроме того, скомпилированная программа может выполняться только на том же типе компьютеров и, как правило, под той же операционной системой, на которую был рассчитан компилятор. Чтобы создать исполняемый файл для машины другого типа, требуется новая компиляция.

Компилируемые языки обычно позволяют получить более быструю и, возможно, более компактную программу, и поэтому применяются для создания часто используемых программ.

Примерами компилируемых языков являются Pascal, C, C++, Erlang, Haskell, Rust, Go, Ada.

Если программа написана на интерпретируемом языке, то интерпретатор непосредственно выполняет (интерпретирует) ее текст без предварительного перевода. При этом программа остается на исходном языке и не может быть запущена без интерпретатора. Можно сказать, что процессор компьютера — это интерпретатор машинного кода. Кратко говоря, интерпретатор переводит на машинный язык прямо во время исполнения программы.

Программы на интерпретируемых языках можно запускать сразу же после изменения, что облегчает разработку. Программа на интерпретируемом языке может быть зачастую запущена на разных типах машин и операционных систем без дополнительных усилий. Однако интерпретируемые программы выполняются заметно медленнее, чем компилируемые, кроме того, они не могут выполняться без дополнительной программы-интерпретатора.

Примерами интерпретируемых языков являются PHP, Perl, Ruby, Python, JavaScript. К интерпретируемым языкам также можно отнести все скриптовые языки.

2. Универсальные и специализированные

Универсальный язык, или язык общего назначения, - язык программирования, ориентированный на решение задач практически из любой области и объединяющий на единой

методической основе наиболее существенные свойства и средства современных машинно- и проблемно-ориентированных языков программирования (например, язык ассемблера, ПЛ/1 и др.).

Специализированные языки чаще всего используются для написания не очень больших программ, поэтому они оптимизированы на быстрое написание программ и уменьшение размера исходного кода, и в меньшей степени на уменьшение ошибок, использования объектно-ориентированное программирования и разделения кода на модули. А универсальные языки, как правило, используются для создания больших и очень больших проектов, поэтому в них все сделано, чтобы уменьшить количество ошибок и облегчить проектирования программ, облегчение разработки крупных программ.

Основное отличие специальных языков от универсальных:

- В них меньше объектно-ориентированных средств и средств доступа технологий COM+, DCOM, CORBA, к функциям API операционных систем;
- 2) Меньше средств многопоточного программирования и распределенного программирования;
- 3) Используются только динамические типы (т. е. тип переменной определяется в зависимости от её значения, а не при объявлении переменной), а не статические.

Единственное исключение: в версии 9 языка Visual FoxPro можно использовать и статические типы переменных.

3. Алгоритмические языки и языки описания данных

Алгоритмические способы программирования умеют описывать данные, но все же используются преимущественно для разработки достаточно крупных, сложных приложений, которые в деталях описывают действия, а именно алгоритмы. В числе таких инструментов: Pascal, Java, C#, C++, иные.

Языки описания данных, как уже можно догадаться из их названия, используются исключительно для описания различного рода данных, используемых в работе приложений. Они применяются в качестве логичного дополнения к алгоритмическим инструментам.

Яркий пример есть в сфере веб-разработки. JavaScript – алгоритмический ЯП, который содержит инструкции для выполнения на стороне клиента. Вдобавок к нему идет синтаксис каскадных таблиц стилей CSS и язык описания данных JSON. Без них скрипты, написанные на JavaScript, не будут работать настолько эффективно.

4. Низкоуровневые и высокоуровневые

Низкоуровневый язык программирования (язык программирования низкого уровня) — язык программирования, близкий к программированию непосредственно в машинных кодах используемого реального или виртуального (например, байт-код, IL) процессора. Для обозначения машинных команд обычно применяется мнемоническое обозначение. Это позволяет запоминать команды не в виде последовательности двоичных нулей и единиц, а в виде осмысленных сокращений слов человеческого языка (обычно английских).

Иногда одно мнемоническое обозначение соответствует целой группе машинных команд, выполняющих одинаковое действие над различными операндами. Кроме машинных команд низкоуровневые языки программирования могут предоставлять дополнительные возможности, такие как макроопределения (макросы). При помощи директив есть возможность управлять процессом трансляции машинных кодов, предоставляя возможность заносить константы и литеральные строки, резервировать память под переменные и размещать исполняемый код по определенным адресам. Часто эти языки позволяют работать вместо конкретных ячеек памяти с переменными.

Как правило, низкоуровневые языки используют особенности конкретного семейства процессоров. Общеизвестный пример низкоуровневого языка — язык ассемблера, хотя правильнее говорить о группе языков ассемблера. Более того, для одного и того же процессора может существовать несколько видов языка ассемблера, совпадающих в машинных командах, но различающихся набором дополнительных функций (директив и макросов).

Также к языкам низкого уровня условно можно причислить CIL, применяемый в платформе Microsoft .NET, Fort

Высокоуровневый язык программирования — язык программирования, разработанный для быстроты и удобства использования программистом. Основная черта высокоуровневых языков — это абстракция, то есть введение смысловых конструкций, кратко описывающих такие структуры данных и операции над ними, описания которых на машинном коде (или другом низкоуровневом языке программирования) очень длинны и сложны для понимания.

Высокоуровневые языки программирования были разработаны для платформенной независимости сути алгоритмов. Зависимость от платформы перекладывается на инструментальные программы — трансляторы, компилирующие текст, написанный на языке высокого уровня, в элементарные машинные команды (инструкции). Поэтому, для каждой платформы разрабатывается платформенно-уникальный транслятор для каждого высокоуровневого языка, например, переводящий текст, написанный на Delphi в элементарные команды микропроцессоров семейства x86[2].

Так, высокоуровневые языки стремятся не только облегчить решение сложных программных задач, но и упростить портирование программного обеспечения. Использование разнообразных трансляторов и интерпретаторов обеспечивает связь программ, написанных при помощи языков высокого уровня, с различными операционными системами программируемыми устройствами и оборудованием, и, в идеале, не требует модификации исходного кода (текста, написанного на высокоуровневом языке) для любой платформы.

Такого рода оторванность высокоуровневых языков от аппаратной реализации компьютера помимо множества плюсов имеет и минусы. В частности, она не позволяет создавать простые и точные инструкции к используемому оборудованию. Программы, написанные на языках высокого уровня, проще для понимания программистом, но менее эффективны, чем их аналоги, создаваемые при помощи низкоуровневых языков. Одним из следствий этого стало добавление поддержки того или иного языка низкого уровня (язык ассемблера) в ряд современных профессиональных высокоуровневых языков программирования.

5. Процедурно-, проблемно- и объектно-ориентированные языки

Проблемно-ориентированный язык – компьютерный язык, разработанный для программирования особого типа задач. Проблемно-ориентированные языки – это формальные языки связи человека с ЭВМ, предназначенные для описания данных (информации) и алгоритмов их обработки (программ) на вычислительной машине. Они определяются спецификой задач, а не особенностями машины.

К числу проблемно-ориентированных алгоритмических языков относятся языки, специально разработанные для описания процессов решения задач некоторого узкого класса. Например, язык Фортран был разработан специально для программирования научных задач. Язык программирования Кобол в основном предназначался для решения экономических задач. Система программирования GPSS ориентирована на моделирование систем с помощью событий. В терминах этого языка легко описывается класс моделей массового обслуживания. Там, где результаты исследований выражаются в терминах времени ожидания, длины очереди, использования ресурсов и т. п., применение языка GPSS полностью оправдано.

Термин “проблемно-ориентированный язык” часто распространяется не только на языки, специально разработанные для определенных задач, но и на все языки высокого уровня.

Процедурно-ориентированные языки программирования относятся к машинно-независимым. Они являются основными языками описания алгоритмов и имеются в математическом обеспечении по существу всех современных вычислительных машин. Операционная система EG позволяет использовать при программировании такие языки, как

[2] x86 - архитектура процессора и одноимённый набор команд, впервые реализованные в процессорах компании Intel. Название образовано от двух цифр, которыми заканчивались названия процессоров Intel ранних моделей — 8086, 80186, 80286 (i286), 80386 (i386), 80486 (i486).

Алгол, Fortran, Cobol и ПЛ-1, относящиеся к этой группе. Будучи почти независимыми от конкретной вычислительной машины, они приближаются по синтаксису к естественным языкам. Они могут лишь косвенно отражать содержание будущей реализации, используя специфические синтаксические конструкции для описания порядка преобразований.

Пользователь применяет процедурно-ориентированные языки программирования (Cobol, Fortran, ПЛ-1) или специализированные языки конкретных банков данных. Непосредственный доступ к данным при обслуживании и использовании банков данных производится с помощью операторов командного языка, включаемых в прикладные программы. Языки программирования, которые могут включать операторы командного языка, называются включающими языками. В связи с тем, что с банком данных взаимодействуют различные пользователи, выделяются физический, логический, системно-логический уровни представления данных.

Объектно-ориентированный язык программирования (ОО-язык) — язык, построенный на принципах объектно-ориентированного программирования.

В основе концепции объектно-ориентированного программирования лежит понятие объекта — некой сущности, которая объединяет в себе поля (данные) и методы (выполняемые объектом действия).

Например, объект человек может иметь поля имя, фамилия и методы есть и спать. Соответственно, в программе можем использовать операторы Человек.Имя:="Иван" и Человек.Есть(пища).

В современных объектно-ориентированных языках используются механизмы:

- Наследование. Создание нового класса объектов путём добавления новых элементов (методов). Некоторые ОО языки позволяют выполнять множественное наследование, то есть объединять в одном классе возможности нескольких других классов.
- Инкапсуляция. Сокрытие деталей реализации, которое позволяет вносить изменения в части программы безболезненно для других её частей, что существенно упрощает сопровождение и модификацию ПО.

Рис. 3 Варианты классификации языков программирования

- Полиморфизм. При полиморфизме некоторые части (методы) родительского класса заменяются новыми, реализующими специфические для данного потомка действия. Таким образом, интерфейс классов остаётся прежним, а реализация методов с одинаковым названием и набором параметров различается. В ООП обычно применяется полиморфизм подтипов (называемый при этом просто «полиморфизмом»), нередко в форме позднего связывания.

Стоит отметить, что процедурно-, проблемно- и объектно-ориентированные языки относятся к машинно-независимым системам программирования.

Трансляторы

Трансляторы бывают двух типов: компиляторы и интерпретаторы. Также существует иная классификация:

- Диалоговый — обеспечивает использование языка программирования в режиме разделения времени
- Синтаксически-ориентированный (синтаксически-управляемый) — получает на вход описание синтаксиса и семантики языка, текст на описанном языке и выполняет трансляцию в соответствии с заданным описанием
- Однопроходной — преобразует исходный код при его однократном последовательном чтении (за один проход)
- Многопроходной — преобразует исходный код после его нескольких чтений (за несколько проходов)
- Оптимизирующий — выполняет оптимизацию создаваемого кода
- Тестовый — получает на вход исходный код и выдающий на выходе изменённый исходный код. Запускается перед основным транслятором для добавления в исходный код отладочных процедур. Например, транслятор с языка ассемблера может выполнять замену макрокоманд на код
- Обратный — выполняет преобразование машинного кода в текст на каком-либо языке программирования

Компилятор

Компилятор – программа, переводящая текст, написанный на каком-то языке программирования в машинный код. Условно говоря, это переводчик исходного кода. Сам процесс преобразования программного кода в машинный называется компиляцией. Компиляция включает в себя:

1. Трансляцию всех модулей программы, написанных на одном или нескольких исходных языках программирования высокого уровня и/или языке ассемблера, в эквивалентные программные модули на низкоуровневом языке, близком машинному коду (абсолютный код, объектный модуль, иногда на язык ассемблера) или непосредственно на машинном языке или ином двоичнокодовом низкоуровневом командном языке
2. Последующую сборку исполняемой машинной программы, в том числе вставка в программу кода всех функций, импортируемых из статических библиотек и/или генерация кода запроса к ОС на загрузку динамических библиотек, из которых программой функции будут вызываться.

Если компилятор генерирует исполняемую машинную программу на машинном языке, то такая программа непосредственно исполняется физической программируемой машиной (например компьютером). В других случаях исполняемая машинная программа выполняется соответствующей виртуальной машиной.

Входная информация для компилятора есть:

1. На фазе трансляции: исходный код программы, являющийся описанием алгоритма или программы на предметно-ориентированном языке программирования;
2. На фазе компоновки: сгенерированные на фазе трансляции файлы объектных кодов модулей программы, а также файлы объектных кодов статических библиотек и данные об используемых динамических библиотеках.

Состав систем программирования

На выходе компилятора — эквивалентное описание алгоритма на машинно-ориентированном языке (объектный код, байт-код).

Компилировать — проводить сборку машинной программы, включая:

1. Трансляцию с предметно-ориентированного языка на машинно-ориентированный язык
2. Компоновка исполняемой машинно-ориентированной программы из сгенерированных на фазе трансляции объектных модулей — модулей, содержащих части кода программы на машинно-ориентированного кода программы.

Интерпретатор

Интерпретатор - второй вид транслятора. В отличие от компиляторов, интерпретаторы не производят машинного кода, они берут исходным текст программы на написанном языке программирования и построчно выполняют его. Этот процесс называется интерпретацией, он выполняется дольше, чем компиляция, за счёт того, что работа программы идёт не напрямую с центральным процессором, а через программу-посредника. Ещё стоит отметить, что компиляторы считывают исходный код только один раз.

Существуют два типа интерпретатора:

1. 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.
2. 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, TCL, Perl, REXX (сохраняется результат парсинга исходного кода), а также в различных СУБД.

Отладчик

Отладчик — компьютерная программа для автоматизации процесса отладки: поиска ошибок в других программах, ядрах операционных систем, SQL-запросах и других видах кода. В зависимости от встроенных возможностей отладчик позволяет выполнять трассировку, отслеживать, устанавливать или изменять значения переменных в процессе выполнения кода, устанавливать и удалять точки останова или условия остановки и т. д.

Основные классы отладчиков: символьные (высокоуровневые, как правило встраиваемые в интегрированные среды разработки) и машинные (низкоуровневые, работающие непосредственно с исполняемым процессорным кодом, включающие дизассемблеры и отладочные символы для представления точек выполнения), однако существует множество вариантов и комбинаций. Существуют специальные классы отладчиков по виду отлаживаемого программного обеспечения (например, отладчики ядра для ядер операционных систем).

Примеры популярных отладчиков: Microsoft Visual Studio Debugger (Windows), Средства отладки Xcode (macOS и iOS), GDB (Linux), Android SDK и Android Studio (Android).

Примеры систем программирования

В данной главе будут рассматриваться наиболее популярные системы программирования ведущих фирм-производителей, таких как Borland/Inprise, Misrosoft. Стоит отметить основные вехи на пути развития систем программирования:

- Переход от одиночных утилит систем программирования к интегрированным диалоговым средам программирования (например, семейство Turbo-продуктов фирмы Borland)
- Развитие инструментальных наборов, расширяющих возможности систем программирования, в частности, в области диалога (разного рода Tool Box)
- Появление объектно-ориентированных диалектов языков C и Pascal; замечу, что, по моему мнению несмотря на то, что Pascal является более строгим и корректным языком, феномен C++ имеет большее значение в силу наличия стандарта
- Возникновение операционной среды Windows со встроенной поддержкой диалога и первых Windows-приложений с помощью SDK (Software Development Keet);
- Создание объектно-ориентированных библиотек, поддерживающих диалоговый режим работы в среде DOS и Windows (TurboVision, Object Windows и MFC)
- Появление систем программирования, облегчающих создание приложений для DOS и Windows
- Развитие механизма встраивания и связывания объектов OLE 2; - Переход к визуальным системам программирования (Visual C++, Delphi, Visual Basic), которые ориентированы на разработку информационных приложений.

Системы программирования фирмы Borland/Inprise

Система программирования Delphi

Система программирования Delphi появилась на рынке в начале 1995 года и быстро завоевала титул первой системы быстрой разработки приложений для Windows, сочетающей в единой среде высокопроизводительный компилятор, визуальные механизмы двунаправленного проектирования и новую методику масштабируемого доступа к базам данных.

Данная среда является одной из ведущих систем программирования, используемых для разработки современных программных продуктов, и в первую очередь приложений операционной системы MS Windows. Система Delphi базируется на использовании языка программирования Object Pascal, который является логическим продолжением и развитием классического языка программирования Паскаль.

Данное название, подобно названию языка Паскаль, также не является случайным. Свое название система программирования Delphi получила в честь существовавшего в древней Греции города Дельфы, где находился знаменитый храм бога Аполлона.

Систему программирования Delphi подобно системе Турбо Паскаль часто называют интегрированной средой программирования. Слово «интегрированный» означает в данном случае, что в системе объединены в одно целое различные средства, способствующие наиболее быстрой и эффективной разработке программы.

К этим средствам относится, во-первых, файловый менеджер, позволяющий не покидая среду программирования создавать новые программные файлы, сохранять их там, где это необходимо и когда необходимо, а также открывать существующие файлы. Во-вторых, это экранный редактор, позволяющий не только набирать и корректировать текст программы, но и в ряде случаев автоматизировать этот процесс и подсказывать программисту те служебные слова, которые можно использовать в данном контексте. В-третьих, — это система отладки программы, которая в большинстве случаев не ограничивается указанием характера сделанной ошибки, указывая также строку, в которой эта ошибка была допущена. В-четвёртых, это разветвленная справочная система, которая содержит не только сведения теоретического характера, но и конкретные примеры разработки приложений в среде Delphi. Все вышеперечисленное далеко не

исчерпывает все многообразие средств, способствующих созданию приложений в данной системе.

Характеризуя среду программирования Delphi, о ней также говорят как о визуальной и событийно-ориентированной. Первое означает, что пользователь визуально, то есть наглядно может увидеть в системе те заготовки, которые в дальнейшем будут использованы для создания экранных объектов в его программе, а затем сам сконструировать ее интерфейс (внешний вид) путем переноса этих заготовок на экранную форму. Второе же означает, что программист может выбрать из имеющегося в системе программирования списка те события, на которые должны реагировать экранные объекты и запрограммировать эту реакцию нужным ему образом.

Существенным дополнением к возможностям обычных систем программирования в системах Delphi является наличие средств подключения и работы с локальными и распределенными системами баз данных. В состав самых первых систем программирования Delphi уже был включен процессор баз данных компании Borland (BDE - Borland Database Engine). Процессор BDE является посредником между прикладными программами и базами данных. Для уменьшения зависимости прикладных программ от конкретной базы данных этот процессор предоставляет пользователям единый интерфейс, благодаря чему при смене базы данных приложение остается вполне работоспособным. В состав процессора BDE входят драйверы систем управления базами данных (СУБД) для некоторых, наиболее распространенных на персональных ЭВМ СУБД: Microsoft Access, FoxPro, Paradox, dBase и некоторых других. В состав BDE входит также драйвер ODBC (Open Database Connectivity), разработанный для включения в системы Delphi возможностей, предоставляемых для связи с базами данных.

Сама компания Borland продолжила развитие собственной системы программирования в части поддержки работы с базами данных. Ею были разработаны технологии IBX (InterBase Express) и dbExpress, которые полностью заменили процессор BDE. В настоящее время компания рекомендует пользоваться не процессором BDE, а более современной технологией dbExpress, которая использует для получения данных исключительно запросы SQL.

Наконец, еще одним важным достоинством системы программирования Delphi является ее универсальность. Дело в том, что многие современные языки и соответствующие системы программирования созданы для решения узкоспециальных задач. Так, язык Cobol предназначен в первую очередь для создания программ в области экономики, язык Fortran - для инженерно-технических расчетов, языки Lisp и Prolog - для работы над системами искусственного интеллекта и т. д. Система же Delphi позволяет создавать профессиональные и эффективно работающие приложения, используемые в самых различных сферах человеческой деятельности. Поэтому время, затраченное будущим специалистом на изучение данной системы программирования, будет потрачено с пользой, вне зависимости от того, какую специализацию он изберет для себя в дальнейшем.

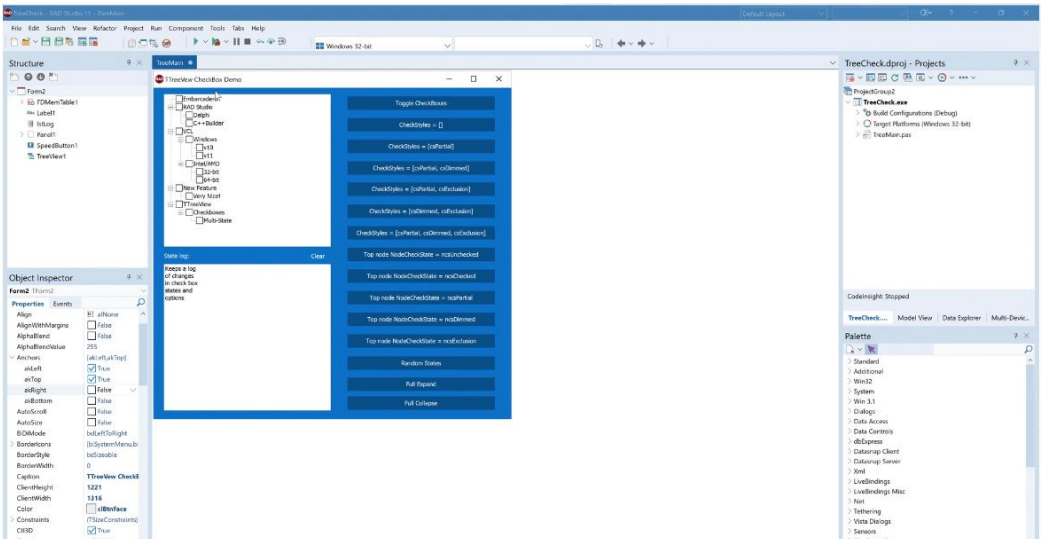

Рис. 4 Система пргораммирования Delphi

Примеры систем программирования

Система программирования C++ Builder.

Язык C++ появился раньше языка Object Pascal и раньше языка Delphi. Именно на примере C++ были продемонстрированы принципы объектно-ориентированного программирования и его достоинства.

Система объектно-ориентированного программирования C++ Builder производства корпорации Borland предназначена для операционных систем Windows. Интегрированная среда C++ Builder обеспечивает скорость визуальной разработки, продуктивность повторно используемых компонент в сочетании с мощью языковых средств C++, усовершенствованными инструментами и разномасштабными средствами доступа к базам данных C++ Builder может быть использован везде, где требуется дополнить существующие приложения расширенным стандартом языка C++, повысить быстродействие и придать пользовательскому интерфейсу качества профессионального уровня.

По своим возможностям C++ Builder практически полностью пересекается с системами Delphi: и здесь и там использован метод технического проектирования программы, называемый визуальным программированием. Отличие от систем Delphi в данном случае заключается в том, что базовым языком данной системы программирования является язык C++.

C++ Builder объединяет в себе комплекс объектных библиотек (STL, VCL, CLX, MFC и др.), компилятор, отладчик, редактор кода и многие другие компоненты. Цикл разработки аналогичен Delphi. Большинство компонентов, разработанных в Delphi, можно использовать и в C++ Builder без модификации, но обратное утверждение не верно.

C++ Builder содержит инструменты, которые при помощи drag-and-drop действительно делают разработку визуальной, упрощает программирование благодаря встроенному WYSIWYG - редактору интерфейса и прочим.

В системе программирования C++ Builder явно прослеживается тенденция построения многоязыковых систем программирования. В большой степени это связано с входящей в состав системы C++ Builder библиотекой визуальных компонентов VCL.

Первоначально эта библиотека была разработана для систем программирования на Паскале, то есть систем Delphi, а позднее была перенесена в C++ Builder. Наличие этой библиотеки в разных системах программирования позволяет пользователю писать программу, состоящую из фрагментов, написанных на разных языках. При этом программист имеет возможность пользоваться одними и теми же абстракциями. В то же время системы Delphi и C++ Builder — это разные системы, поэтому реально создавать многоязыковые программы с их помощью нелегко.

Библиотека VCL замечательна еще и тем, что она полностью построена на принципах объектно-ориентированного программирования и единой иерархии классов с общим базовым классом TObject, находящимся в основе этой иерархии. Все классы VCL являются потомками этого класса. Наличие общего корня библиотеки классов позволяет использовать полиморфизм для реализации общих алгоритмов и структур данных. По своей функциональности библиотека VCL в значительной степени пересекается с другими широко распространенными

библиотекамиС++, в частности, со стандартной библиотекой С++, в том числе со стандартной библиотекой шаблонов STL.

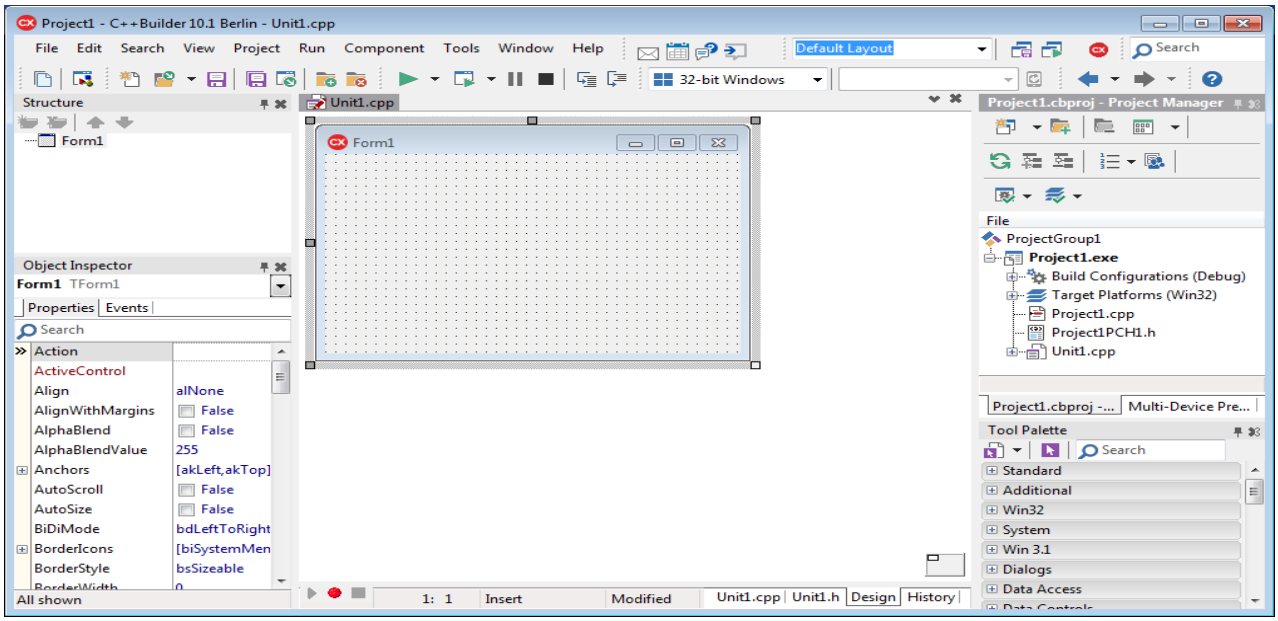

Рис. 5 Система программирования C++ Builder

Системы программирования фирмы Microsoft

К наиболее распространенным системам программирования для настольных ЭВМ относятся системы, выпускаемые компанией Microsoft. Весь комплекс программ, поставляемых компанией Microsoft, следует называть единой операционной средой, предназначенной для разработчиков системного и прикладного программного обеспечения.

Системы, выпускаемые компанией Microsoft, выполнены в едином стиле, их интерфейс хорошо продуман. Многооконный интерфейс позволяет одновременно видеть различную информацию о создаваемой, тестируемой или исполняемой программе. Все системы имеют развитые отладчики, которые работают в терминах базового языка программирования (Basic/C++/Язык ассемблера). В любой момент времени у программиста есть возможность проверить состояние того или иного объекта данных, а в процессе отладки можно даже менять некоторые значения переменных и сразу продолжать работу с точки остановки программы без дополнительной перекомпиляции.

Система программирования Visual Basic.

Microsoft Visual Basic - сегодня самая популярная в мире система проектирования приложений для Windows. Среда Visual Basic может с успехом использоваться начинающими пользователями для познания секретов программирования и увлекательных занятий по созданию несложных для начала приложений и, в то же время, предоставляет мощные инструменты разработки опытным программистам. Чрезвычайно развитые справочная система, средства обучения, мастера и программы-надстройки позволяют при построении приложения и работе в Visual Basic найти выход из любой ситуации и получить ответ на любой вопрос. Начинать работать с Visual Basic можно практически с любым уровнем подготовки.

Язык Basic, в том виде, каким он предстает в современных системах программирования, сильно отличается от своей первоначальной версии. В настоящее время это объектно-ориентированный язык, обладающий всеми возможностями других, более новых языков программирования, но оставшийся весьма простым для изучения, благодаря простым изобразительным средствам. Процесс создания диалоговых форм и расстановки на них элементов управления диалогом благодаря визуальному подходу стал несложным и понятным. Система программирования в процессе создания форм автоматически создает программу на

языке Visual Basic. Отладчик, встроенный в систему программирования, работает в терминах языка Visual Basic, поэтому отладка программ не представляет особой сложности.

В целом, систему Visual Basic можно определить, как инструментальную среду для разработки самых различных программных продуктов. Создаваемые в этой интегрированной инструментальной среде программы обладают свойством автономности и в состоянии после завершения разработки функционировать в отрыве от самой среды. Следует только помнить о необходимости сопровождать распространение программы, написанной в системе Visual Basic, библиотеками, отслеживая совместимость версий стандартных библиотек фирмы Microsoft с версией созданной программы. Отсутствие нужной библиотеки, а иногда и небольшого системного файла в системном каталоге неминуемо заблокирует работу программы.

Если задаться вопросом - что такое Visual Basic - компилятор или интерпретатор, можно смело сказать: «И то, и другое». Его нельзя всецело отнести ни к компиляторам, ни к интерпретаторам.

Основным признаком интерпретатора Visual Basic является то, что созданные с помощью него программы выполняются только в среде разработки. Программу можно запустить непосредственно из среды и, если в ней есть ошибки, они сразу же распознаются. Все это наблюдается и в Visual Basic, где можно запустить приложение непосредственно в среде программирования. При этом Visual Basic использует технологию Threaded-p-Code, при которой каждая написанная строка кода преобразуется в промежуточный код - Threaded-p-Code. Это не машинный код, но такой код выполняется быстрее, чем при работе с обычным интерпретатором. Во-первых, Visual Basic сразу же проверяет синтаксис программы и выдает сообщение, если присутствует ошибка. Также можно самим искать эти ошибки.

Но при этом Visual Basic - не просто интерпретатор, так как это означало бы, что приложения выполняются только в среде Visual Basic. Эта среда программирования предоставляет возможность создавать и исполняемые EXE-файлы, поэтому она относится и к компиляторам. Basic нельзя назвать чистым компилятором, так как в отличие, например, от Visual C++, Visual Basic не создает исполняемый файл сразу же при запуске из среды разработки. Для создания такого файла необходимо сделать это явно (команда File\Make ***.EXE). Начиная с пятой версии, Visual Basic обладает так называемым «Native Compiler», то есть компилятором, который может создавать машинный код. Таким образом. Visual Basic объединяет в себе возможности, как интерпретатора, так и компилятора. И это имеет больше преимуществ, чем недостатков.

Система программирования Visual C++.

Система программирования Microsoft Visual C++ представляет собой реализацию среды разработки для распространенного языка системного программирования C++, выполненную компанией Microsoft. Эта система программирования в настоящее время построена в виде интегрированной среды разработки, включающей в себя все необходимые средства для разработки результирующих программ, ориентированных на выполнение под управлением ОС типа Microsoft Windows различных версий.

Возможность использовать язык C++ превращает эту систему программирования в инструмент, позволяющий создавать не только обычные офисные приложения, но и решать другие задачи.

Основу системы программирования Microsoft Visual C++ составляет библиотека классов MFC (Microsoft foundation classes). В этой библиотеке реализованы в виде классов C++ все основные органы управления и интерфейса ОС. Также в ее состав входят классы, обеспечивающие разработку приложений для архитектуры клиент-сервер и трехуровневой архитектуры (в современных версиях библиотеки). Система программирования Microsoft Visual C++ позволяет разрабатывать любые приложения, выполняющиеся в среде ОС типа Microsoft Windows, в том числе серверные или клиентские результирующие программы, осуществляющие взаимодействие между собой по одной из указанных выше архитектур.

Классы библиотеки MFC ориентированы на использование технологий COM/DCOM, а также построенной на их основе технологии ActiveX для организации взаимодействия между клиентской и серверной частью разрабатываемых приложений. На основе классов библиотеки пользователь может создавать свои собственные классы в языке C++, организовывать свои структуры данных.

В отличие от систем программирования компании Borland, система программирования Microsoft Visual C++ ориентирована на использование стандартных средств хранения и обработки ресурсов интерфейса пользователя в ОС Windows.

Система программирования Microsoft Visual C++ выдержала несколько реализаций. В процессе выхода новых версий системы программирования было выпущено и несколько версий библиотеки MFC, на которой основана данная система.

Сама по себе библиотека MFC является, по мнению автора, довольно удачной реализацией широкого набора классов языка C++, ориентированного на разработку результирующих программ, выполняющихся под управлением ОС типа Microsoft Windows. Это во многом обусловлено тем, что создатель библиотеки компания Microsoft одновременно является и создателем ОС типа Microsoft Windows, на которые ориентирован объектный код библиотеки. Библиотека может быть подключена к результирующей программе с помощью обычного компоновщика, либо использоваться как динамическая библиотека, подключаемая к программе во время ее выполнения. Библиотека MFC достаточно широко распространена. Ее возможно использовать не только в составе систем программирования производства компании Microsoft, но и в системах программирования других производителей.

В систему программирования встроен удобный интерактивный отладчик, работающий в терминах языка C++ или языка ассемблера и позволяющий одновременно видеть на экране тексты различных фрагментов программ, значения переменных и регистров центрального процессора ЭВМ, стек вызовов процедур и другую необходимую при отладке информацию. Отладчик позволяет менять значения переменных, что иногда помогает программисту проверить гипотезу о причинах неправильного поведения программы, а впоследствии и исправить программу.

При работе в системе Visual C++ доступна вся справочная информация, как о самой системе, так и о языке C++, библиотечных функциях и операционной системе Windows. Справочник снабжен большим количеством примеров, которые часто позволяют повысить

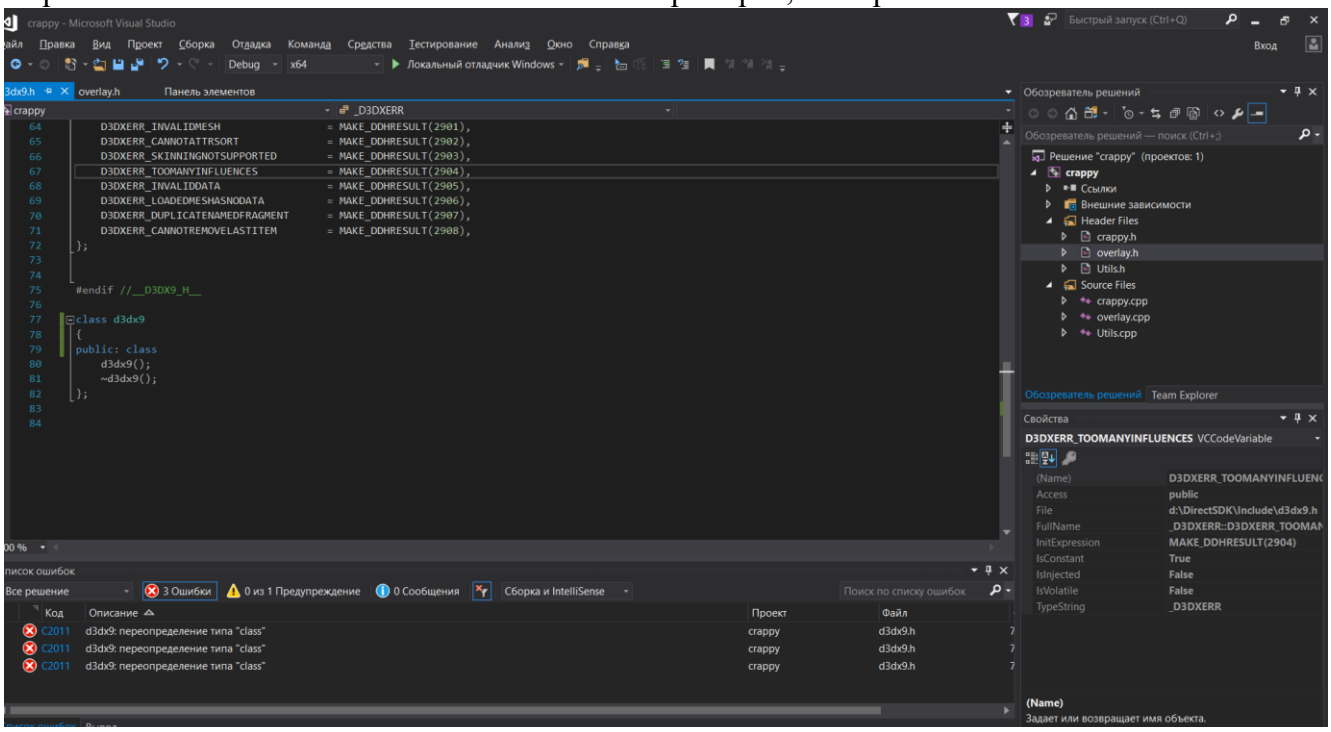

Рис. 6 Внешний вид системы программирования Microsoft Visual C++

эффективность как процесса программирования, так и процесса работы уже подготовленной программы.

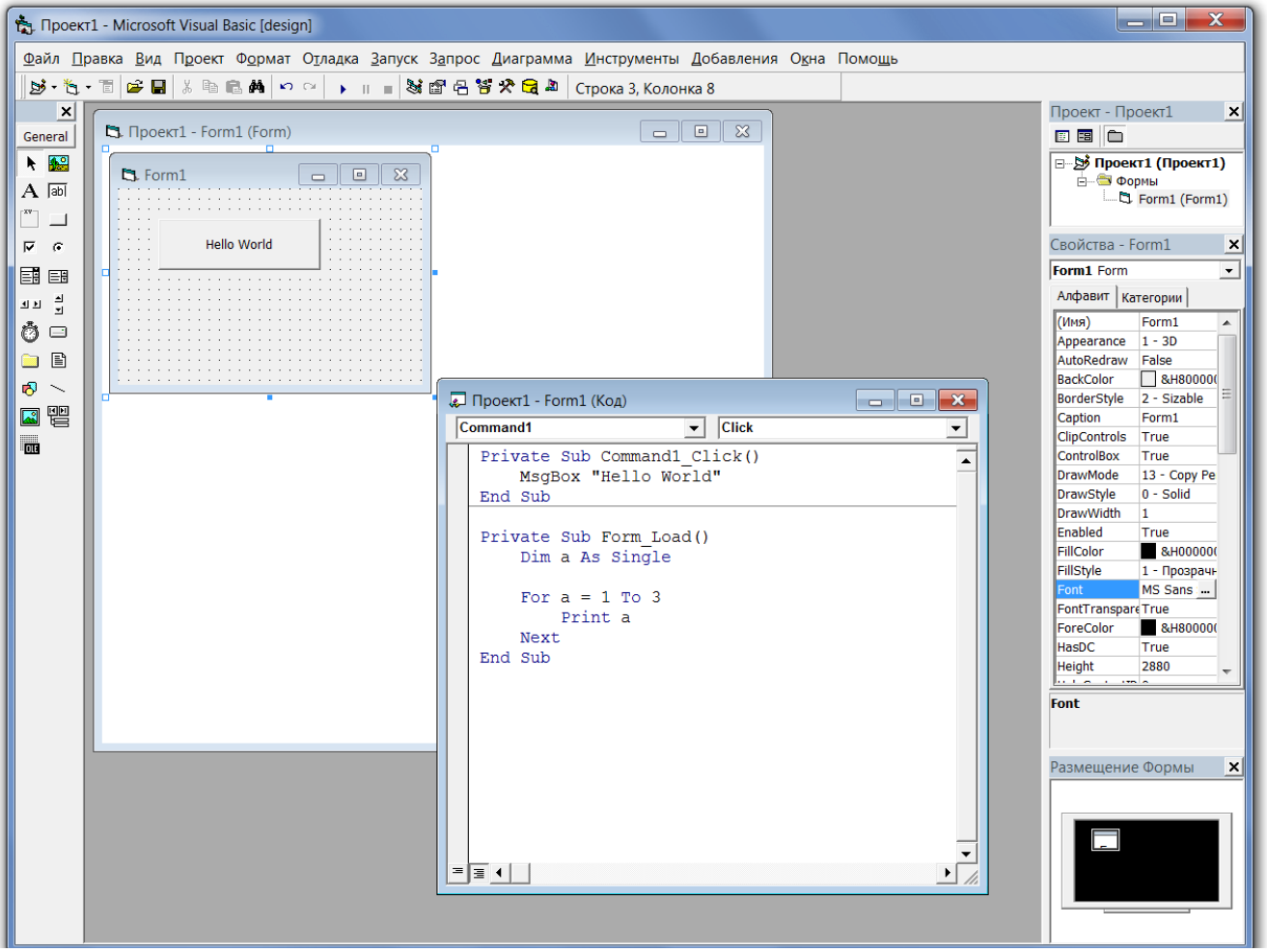

Рис. 7 Внешний вид системы программирования Microsoft Visual Basic (версии 6.0)

Заключение

Системы программирования позволяют избежать большого объема однообразных действий и тем самым существенно повысить эффективность процесса разработки и отладки, то есть они являются RAD[3]-средами различной степени автоматизации процесса программирования.

Работа в системе программирования даёт программисту:

- Возможность использования встроенного многофайлового текстового редактора, специально ориентированного на работу с исходными текстами программ;
- Иметь автоматическую диагностику выявленных при компиляции ошибок, когда исходный текст программы, доступный редактированию, выводится одновременно с диагностикой в многооконном режиме;
- Возможность параллельной работы над несколькими проектами. Менеджер проектов позволяет использовать любой проект в качестве шаблона для вновь создаваемого проекта;
- Минимум перекомпиляции. Ей подвергаются только редактировавшиеся модули;

[3] RAD (от англ. rapid application development — быстрая разработка приложений) — концепция организации технологического процесса разработки программных продуктов, ориентированная на максимально быстрое получение результата в условиях сильных ограничений по срокам и бюджету и нечётко определённых требований к продукту.

- Возможность загрузки отлаживаемой программы в имеющиеся средства отладки, и возможность работы с ними без выхода из оболочки;
- Возможность подключения к оболочке практически любых программных средств.

В последнее время, функции интегрированных сред разработки становятся стандартной принадлежностью программных интерфейсов эмуляторов и отладчиков-симуляторов.

Подобные функциональные возможности, в сочетании с дружественным интерфейсом, в состоянии существенно увеличить скорость разработки программ, особенно для микроконтроллеров и процессоров цифровой обработки сигналов, являющихся очень трудоёмкими и труднообозримыми процессами.

Предметный указатель

Список иллюстраций