ИСТОРИЯ ПРОГРАММИРОВАНИЯ

Лекция № 1

Программирование как направление деятельности человека возникло одновременно с появлением первых вычислительных машин. Поэтому историю развития программирования нужно рассматривать параллельно с историей развития самих компьютеров.

У «программиста» эпохи фон Неймана, что у его сегодняшнего современника основная цель – подчинить машину желаниям человека, заставить ее выполнять нужные человеку действия. С течением времени разнились, менялись, эволюционировали только лишь способы достижения этой цели.

ИСТОРИЯ ПРОГРАММИРОВАНИЯ

Основные этапы развития ЭВМ

I этап (1940-е – середина 50-х)

Точкой, с которой обычно начинают отсчитывать этапы развития ЭВМ, является Вторая Мировая Война. Потребность в автоматизации вычислений (баллистики, криптографии и т.д.) была так велика, что над созданием машин типа построенных Эйкеном и Цузе трудились одновременно сразу несколько групп исследователей.

Начиная с 1943 г. группа специалистов под руководством Джона Мосли и Пресперта Экерта в США начала конструировать ЭВМ на основе электронно-вакуумных ламп. Предыдущие же модели типа эйкеновской основывались на реле. Как результат, созданная ими машина ENIAC (1946 г.) работала в 1000 раз быстрее, чем Марк-1. Однако перед вычислениями машину приходилось долго готовить, на это уходило от нескольких часов до нескольких дней. Для упрощения процесса задания программ решено было сконструировать новую машину, которая могла бы хранить программу в своей памяти.

ПРОГРАММИРОВАНИЯ

В процессе создания ENIAC в 1945 году к работе был привлечен известный американский математик Джон фон Нейман, который вскоре сумел сформулировать пять основных принципов функционирования универсальных вычислительных устройств:

1. Любую ЭВМ образуют:
   a. арифметико-логическое устройство, выполняющее арифметические и логические операции;
   b. устройство управления, «дирижирующее» операциями;
   c. запоминающее устройство, или память для хранения программ и данных;
   d. внешние устройства ввода-вывода информации.

2. ЭВМ работает в двоичной системе счисления;

ПРОГРАММИРОВАНИЯ

3. Принцип программного управления: устройство управления автоматически выполняет команды, которые записаны в памяти. Команды выполняются друг за другом в определенной последовательности.

4. Принцип однородности памяти: программы и данные хранятся в одной и той же памяти. Над командами программы можно выполнять те же действия, что и над данными.

5. Принцип адресности: каждая ячейка памяти имеет свой адрес (номер).

ПРОГРАММИРОВАНИЯ

3. Принцип программного управления: устройство управления автоматически выполняет команды, которые записаны в памяти. Команды выполняются друг за другом в определенной последовательности.

4. Принцип однородности памяти: программы и данные хранятся в одной и той же памяти. Над командами программы можно выполнять те же действия, что и над данными.

5. Принцип адресности: каждая ячейка памяти имеет свой адрес (номер).

На этих принципах, как на фундаменте, основывались в дальнейшем все последующие поколения машин.

ПРОГРАММИРОВАНИЯ

Третий пункт, принцип программного управления. Именно из этого принципа вытекает определение программы в ее классическом понимании – как последовательного набора команд, выполняемых процессором.

Отсюда возникает и необходимость в подготовке специально обученных людей, которые бы знали, какие команды и в какой последовательности нужно применять для решения определенной задачи.

Программирование машин становится направлением деятельности человека.

ПРОГРАММИРОВАНИЯ

Для первых ЭВМ, созданных по принципам фон Неймана, например, машины Мориса Уилкса (1949 г.) программы приходилось писать на машинном языке, то есть в кодах, непосредственно воспринимаемых компьютером. Это, конечно, было гораздо лучше и быстрее, но все же еще оставалось очень тяжелым, малопроизводительным и кропотливым занятием, в ходе которого легко было ошибиться.

Решением стало изобретение т.н. ассемблеров, или автокодов. **Ассемблеры – языки символического кодирования. Для каждого типа ЭВМ существует свой язык символического кодирования, так как его структура, обозначения, операции и т.д. определяются структурой конкретной ЭВМ.**

ПРОГРАММИРОВАНИЯ

Ассемблеры недалеко шагнули от машинных команд: просто вместо непонятных кодов стали использоваться мнемонические (легко запоминаемые) обозначения. Например, если 07FF – команда сложения на машинном языке, то в ассемблере она представляется мнемокодом ADD. Вследствие этого, программы на ассемблере очень легко переводятся в машинные коды при помощи специальной программы, называемой также ассемблером, и занимают крайне мало места (в единицах памяти, но не по размеру самого текста программы). Они обеспечивают большое быстродействие и наиболее полно учитывают особенности компьютера.

Подводя итог: программирование на данном этапе – трудоемкий процесс, диктующий необходимость знания всех команд ЭВМ и ее структуры, с непосредственной работой математика-программиста за пультом машины.

ИСТОРИЯ ПРОГРАММИРОВАНИЯ

II этап (середина 50-х – середина 60-х)

В 1948 г. был изобретен транзистор. К середине 50-х были найдены очень дешевые способы производства транзисторов, а во второй половине 50-х появились первые транзисторные ЭВМ. Они были в сотни раз меньше ламповых при той же производительности. Первой из них считается модель RCA-501 (1959 г., США).

К середине 60-х появились и более компактные версии внешних устройств. Все это позволило 1965 г. фирме Digital Equipment выпустить первый мини-компьютер PDP-8. Он поступил в продажу по цене в 20 тыс. долларов. Правда, размером он был с холодильник, но по сравнению со шкафами и даже комнатами, занимаемыми ламповыми ЭВМ, смотрелся довольно привлекательно.

В СССР к этому поколению относятся серии «Минск» и БЭСМ, наиболее совершенная из них – БЭСМ-6.

ПРОГРАММИРОВАНИЯ

На этом этапе были созданы и более совершенные языки программирования, т.н. языки высокого уровня. Такой язык – это набор символов и терминов, который в соответствии с правилами синтаксиса описывает алгоритм решения задачи. Своими конструкциями и правилами написания этот язык, с одной стороны, близок к математическому выражению задачи, а с другой стороны, содержит такие выражения, которые близки к естественному языку, чаще всего английскому. **Программы, написанные на таких языках, либо преобразуются в машинные команды (это делается с помощью специальных программ, называемых трансляторами или компиляторами) или выполняются построчно (интерпретируются) при помощи программ-интерпретаторов.**

Первый коммерчески используемый язык программирования высокого уровня FORTRAN (FORmula TRANslator – транслятор формул) был разработан в 1958 г. фирмой IBM под руководством Джона Бэкуса. Он предназначен, прежде всего, для научных вычислений и очень близок по форме к записи математических формул. У Фортрана есть один существенный недостаток: он не обеспечивает надежности программирования (защиты от ошибок) из-за своей громоздкости и несовершенства логических возможностей [2].

ПРОГРАММИРОВАНИЯ

Другим языком высокого уровня стал Алгол-60 (ALGOrithmic Language – алгоритмический язык), который появился в 1960 г. Это более совершенный язык, чем Фортран; он обладает большей гибкостью при проектировании программ, обеспечивает более высокую надежность программирования. В основном использовался в советских ЭВМ.

Именно после создания Алгола началось бурное развитие языков программирования.

Также заслуживающим внимания языком стал разработанный в конце 50-х годов LISP (LISt Processing – обработка списков). Он используется и по сей день для программирования в области искусственного интеллекта.

Языки высокого уровня позволили значительно упростить процесс написания программ, так как они ориентированы на удобство описания решаемых с их помощью задач, а не на особенности конкретного компьютера. Разумеется, для каждой программы на языке высокого уровня программист может написать более компактную и быстродействующую программу на ассемблере, но из-за трудоемкости процесса обычно игра не стоит свеч.

ПРОГРАММИРОВАНИЯ

Языки высокого уровня и на сегодняшний момент являются основными инструментами, при помощи которых программисты составляют программы.

Из-за введения понятия “языки высокого уровня”, машинно-ориентированные языки (ассемблеры) стали называть языками низкого уровня.

Итог: появились более совершенные алгоритмические языки, теперь не требовалось присутствия программистов в зале, решении задач стало проводиться в пакетном режиме, а не по-командно, как это было на предыдущем этапе.

ПРОГРАММИРОВАНИЯ

III этап (середина 60-х – середина 70-х)

На предыдущем этапе развития ЭВМ транзисторы изготовлялись по отдельности, и при сборке их приходилось соединять и спаивать вручную. В 1958 г. Джек Килби придумал, как на одной пластине полупроводника получить несколько транзисторов.

В 1959 г. Роберт Нойс (будущий основатель фирмы Intel) изобрел более совершенный метод, позволивший создавать на одной пластине и транзисторы, и все необходимые соединения между ними. Полученные электронные схемы стали именоваться интегральными схемами, или чипами.

Первая ЭВМ, созданная на чипах – результат работы американской фирмы IBM, IBM-360 (1965 г.).

В СССР в данное поколение вошли семейства машин ЕС (1022, 1035, 1060) и СМ (2, 3, 4, 1420).

ПРОГРАММИРОВАНИЯ

В 1970 г. фирма Intel начала продавать интегральные схемы памяти. В том же году произошло еще одно знаменательное событие – та же Intel сконструировала интегральную схему, аналогичную по своим функциям центральному процессору большой ЭВМ.

Так появился первый микропроцессор Intel-4004. В последующие годы выходили модернизированные версии этой схемы, и апогеем стал микропроцессор Intel-8080 (1974 г.), который до конца 70-х оставался стандартом для микрокомпьютерной индустрии.

Здесь начал применяться т.н. принцип программной совместимости – когда программы, разработанные для одной ЭВМ, могут быть использованы для другой ЭВМ этой серии. Каждая новая модель семейства обладает более мощными техническими возможностями, чем предыдущие.

ПРОГРАММИРОВАНИЯ

В 1975 г. появился первый коммерчески распространяемый компьютер на основе Intel-8080 под названием Альтаир-8800. Он был выпущен фирмой MITS. Несмотря на довольно ограниченные возможности Альтаира, его появление было встречено с энтузиазмом, и в первые месяцы было продано несколько тысяч комплектов машины.

В конце 1975 г. Пол Ален и Билл Гейтс (будущие основатели фирмы Microsoft) создали для Альтаира интерпретатор языка Basic, позволявшего пользователям достаточно просто общаться с компьютером и писать для него программы.

Можно сказать, что именно Альтаир-8800 стал первым персональным компьютером.

С этого момента интерес к компьютерам среди широкой массы людей стал расти в геометрической прогрессии.

ПРОГРАММИРОВАНИЯ

Этот этап знаменателен также появлением нескольких новых ЯВУ. Первым стал Паскаль, созданный в 1970 г. Он был создан ведущим в то время специалистом в области программирования Н.Виртом. Это очень простой и компактный язык, его понятия близки к фундаментальным понятиям математики. Благодаря этому он завоевал свою популярность, и даже по сей день используется для обучения специалистов программированию как наиболее полно отражающий базовые конструкции любой разновидности ЯВУ.

Другой новинкой стал упомянутый ранее Basic (Beginner's All-purpose Symbolic Instruction Code – универсальный код символических инструкций для начинающих). Первоначально он был разработан Дж. Келени и Т. Курцем из Дартмутского колледжа (США) в 1965 г. и предназначался для вводного курса по информатике. Однако благодаря своей простоте и возможности работы в диалоговом режиме этот язык быстро нашел призвание среди пользователей-непрограммистов.

ПРОГРАММИРОВАНИЯ

Важную роль в дальнейшем сыграет изобретенный в 1972 г. Денисом Ричи и Кеном Томпсоном из Bell Laboratories язык Си. Он похож на Паскаль, но исправляет его некоторые ошибки и содержит новые элементы. Си разрабатывался как мощный и гибкий язык. К сожалению, он не так легко читается как Паскаль, и предназначен скорее для корпоративных приложений. Тем не менее, пользователям он понравился, и был сделан переход от Паскаля к Си. Это ознаменовало собой переход от языков предыдущего поколения к сегодняшним языкам программирования. **Также следует отметить, что Си использовался для разработки многих операционных систем в дальнейшем, так как сам разрабатывался для только что появившейся в то время системы UNIX.**

Си и Паскаль относятся к структурным языкам программирования, тогда как предыдущие – к функциональным.

ПРОГРАММИРОВАНИЯ

IV этап (середина 70-х – настоящее время)

На этом этапе на смену чипам приходят БИС – большие интегральные схемы. Стали появляться многопроцессорные вычислительные системы такие, как B-7770 фирмы Burroughs, ILLIAC IV Иллинойского университета или советский «Эльбрус-2». Также **стали производиться дешевые и компактные мини- и персональные ЭВМ, а на их основе формироваться вычислительные сети.**

Успех Альтаир-8800 заставил многие фирмы также заняться производством компьютеров. Нарастающая конкуренция подталкивала производителей искать всё лучшие и лучшие решения. Компьютеры стали продаваться в полной комплектации, с мониторами и внешними устройствами, спрос на них стал составлять десятки, а затем и сотни тысяч штук в год.

ПРОГРАММИРОВАНИЯ

Ажиотаж продажи компьютеров привлек внимание фирмы IBM. Было принято решение создать собственный персональный компьютер IBM, и для решения этой задачи было сформировано специальное подразделение. Однако из-за недооценки руководством важности такого исследования денежных ресурсов было выделено не так много, и поэтому был дан «зеленый свет» на использование комплектующих, изготовленных другими производителями. И это сыграло только на руку исследователям: они **выбрали новейший в то время процессор Intel-8088, значительно превосходивший все существовавшие тогда аналоги, а разработка программного обеспечения была поручено небольшой фирме Microsoft.**

Результатом такой кооперации стало появление в августе 1981 г. нового компьютера IBM PC. Вскоре он приобрел большую популярность у пользователей, а через год – занял ведущее место на рынке, потеснив всех конкурентов.

Фактически IBM PC стал стандартом персонального компьютера.

ИСТОРИЯ ПРОГРАММИРОВАНИЯ

В области языков программирования любопытных новинок тоже появилось немало. Правда, следует уточнить, что эти новинки были скорее продолжениями идей предыдущего этапа.

Для начала отметим такой язык, как ADA. Этот ЯВУ был разработан по заказу Министерства обороны США в 1979 г. и назван так в честь первой программистки Ады Лавлейс, дочери Дж. Байрона, которая еще в XIX веке создавала первые программы для машины Бэббиджа. В этом языке использовано много идей из Паскаля и Алгола-68, а также заимствованы лучшие конструкции других языков.

ADA является представителем модульных языков программирования.

ПРОГРАММИРОВАНИЯ

Бьярн Страуструп оценил этот подход и на его основе разработал язык, получивший название «Си с классами».

Вскоре Си с классами превратился в отдельный язык C++ (Си плюс плюс), который был представлен общественности в 1983 г.

Классы – это такие программные абстракции, которые представляют собой как бы шаблон или тип объекта. Поэтому говорят, что объект является экземпляром того или иного класса. Это позволяет программировать любые объекты реального мира, вкупе со всеми их атрибутами (свойствами) и поведением (функциями, или методами).

Ранее, в структурном программировании, приходилось работать с ограниченным набором типов данных, которые ограничивали уровень абстракции и были не очень удобны для представления объектов реального мира.

C++ был разработан как модификация Си при помощи ООП, при этом с сохранением скорости, присущей Си и возможности выполняться на различных платформах (видах компьютеров).

C++ часто используют для создания различных симуляций, в частности, компьютерных игр.

ПРОГРАММИРОВАНИЯ

Язык Java.

В начале 90-х интерактивное телевидение казалось технологией будущего. Sun Microsystems решила, что для интерактивного ТВ понадобится специальный, хорошо переносимый (с одного вида компьютеров на другой) язык программирования. Этим языком стал Java (Джава – сорт кофе; в данной технологии вообще много названий, связанных с этим напитком). В 1994 команде разработчиков пришлось сменить направление разработки – интерактивное ТВ оказалось пустой мечтой. Но Всемирная Сеть росла и развивалась. Решено было использовать Java как веб-язык.

В 1995 году компания Netscape лицензирует Java для создания программы-броузера Сети. Когда броузер увидел свет, Java показался языком будущего – настолько все было привлекательно и ново. Однако и у него оказались **недостатки: проблемы с оптимизацией кода и, следовательно, с быстродействием, упрощение в сравнении с C++**, что оттолкнуло от него некоторых серьезных программистов как от «детского языка». Несмотря на это, у Java есть и несомненные **плюсы: использование ООП, поддержка интерфейсов, переносимость, автоматическая сборка «мусора», бОльшая простота изучения в сравнении с C++**.

ПРОГРАММИРОВАНИЯ

Таким образом можно сказать, что **Java и C++ являются на данный момент основными языками для разработки приложений корпоративного уровня.**

Еще одним языком **помимо Java, который используется в Сети, является Perl и PHP. Perl был разработан Ларри Воллом в 1987 как замена устаревающим утилитам UNIX, использовавшихся для операции с текстом.** Как следствие, Perl (PHP) имеет очень сильные функции анализа текста. Поэтому он, в основном, **применяется для создания ядра веб-интерфейсов или в скриптах, которые модифицируют файлы настроек сайта, где анализ текста является важным моментом и не только для этого.**

ПРОГРАММИРОВАНИЯ

Немного другое направление развила **фирма Microsoft. Она расширила идеи языка BASIC, выпустив Visual Basic (VB). VB, как и его прародитель – своего рода язык для непрограммистов, хотя при желании на нем можно составлять и комплексные программы.**

Основная цель VB – без лишних усилий позволить пользователям создавать программы с мощными экранными интерфейсами. Интерфейсы строятся здесь из специальных компонентов, называемых widget. Это могут быть различные меню, картинки, ползунки, иконки и пр. У виджетов есть набор свойств (например, цвет) и событий (например, нажатие на него левой кнопкой мыши), что является основой любого интерфейса пользователя в современной системе.

ПРОГРАММИРОВАНИЯ

V этап (?? – ??) Просто почитайте:

Возможно, новый этап развития машин будет связан с нано-технологиями: например, компания Intel недавно сообщила, что ее инженеры работают над процессорами размером 15, 10, 7 и 5 нанометра. Возможность применения подобных процессоров граничит с фантастикой – быть может, когда-нибудь нам доведется увидеть живую клетку с внедренным в нее процессором? А может быть даже запрограммировать ее?

А вспомним тот факт, что белкИ, как говорят ученые, составляют в геноме определенный код. Какие тут открываются возможности для будущих программистов…

Другой областью, которая может вызвать скачок в развитии машин, является искусственный интеллект. Одно из толкований целей этой науки состоит как раз в утверждении, что она должна создавать методы автоматического решения задач, считающихся в человеческом понимании интеллектуальными. В частности, это задача разработки средств автоматического выполнения функций алгоритмиста и программиста, то есть интеллектуальных функций по формализации задач и составлению программ для их решения. Вполне может оказаться, что машины научатся думать за нас, и программисты будут больше не нужны. Человеческий фактор исчезнет. Многие фантасты уже не раз рисовали печальные последствия такого хода событий… Но человеку свойственно всегда с опаской относится к новому, чуждому, неизведанному. Быть может, именно следующий этап развития машин и программирования даст и всему человечеству возможность перейти на качественно новый уровень.

ОБЩИЕ СВЕДЕНИЯ

ЛЕКЦИЯ № 2

Си и Паскаль относятся к структурным языкам программирования, тогда как предыдущие – к функциональным.

Структурное программирование – методология программирования, базирующаяся на системном подходе к анализу, проектированию и реализации программного обеспечения.

Структурное программирование предполагает точно обозначенные управляющие структуры, программные блоки, отсутствие инструкций безусловного перехода (GOTO), автономные подпрограммы, поддержку рекурсии и локальных переменных.

Реку́рсия — процесс повторения элементов самоподобным образом. Например, если два зеркала установить друг напротив друга, то возникающие в них вложенные отражения суть одна из форм бесконечной рекурсии. Термин «*рекурсия*» используется в различных специальных областях знаний — от лингвистики до логики, но наиболее широкое применение находит в математике и информатике.

ОБЩИЕ СВЕДЕНИЯ

Основу структурного программирования составляют следующие положения:

- **Сложная задача разбивается на более мелкие, функционально лучше управляемые задачи. Каждая задача имеет один вход и один выход [8]. В этом случае управляющий поток программы состоит из совокупности элементарных подзадач с ясным функциональным назначением.**

ОБЩИЕ СВЕДЕНИЯ

- **Простота управляющих структур, используемых в задаче. Это положение означает, что логически задача должна состоять из минимальной, функционально полной совокупности достаточно простых управляющих структур.**

ОБЩИЕ СВЕДЕНИЯ

- **Разработка программы должна вестись поэтапно. На каждом этапе должно решаться ограниченное число четко поставленных задач с ясным пониманием их значения и роли в контексте всей задачи. Если такого понимания не достигается, это говорит о том, что данный этап слишком велик и его нужно разделить на более мелкие шаги.**

ОБЩИЕ СВЕДЕНИЯ

Концепцию модульного программирования можно сформулировать в виде нескольких ключевых понятий и положений:

- **Функциональная декомпозиция задачи – разбиение большой задачи на ряд более мелких, функционально самостоятельных подзадач – модулей. Модули связаны между собой только по входным и выходным данным.**
- **Модуль – основа концепции модульного программирования. Каждый модуль в функциональной декомпозиции представляет собой «черный ящик» с одним входом и одним выходом.**

ОБЩИЕ СВЕДЕНИЯ

- **Реализуемые решения должны быть простыми и ясными. Если назначение модуля непонятно, то это говорит о том, что декомпозиция начальной или промежуточной задачи была проведена недостаточно качественно. В этом случае необходимо еще раз проанализировать задачу и, возможно, провести дополнительное разбиение на подзадачи. При наличии сложных мест в проекте их нужно подробно задокументировать с помощью продуманной системы комментариев. Этот процесс нужно продолжать до тех пор, пока действительно не будет достигнуто ясного понимания назначения всех модулей задачи и их оптимального сочетания.**

ОБЩИЕ СВЕДЕНИЯ

Модульный подход позволяет безболезненно производить модернизацию программы в процессе ее эксплуатации и облегчает ее сопровождение.

Дополнительно модульный подход позволяет разрабатывать части программ одного проекта на разных языках программирования, после чего с помощью компоновочных средств объединять их в единый загрузочный модуль.

ОБЩИЕ СВЕДЕНИЯ

Метод программирования – объектно-ориентированное программирование (ООП).

Идея ООП заключается в стремлении связать данные с обрабатывающими эти данные процедурами в единое целое – объект. ООП основано на трех важнейших принципах, придающих объектам новые свойства. Этими принципами являются инкапсуляция, наследование и полиморфизм.

ОБЩИЕ СВЕДЕНИЯ

Инкапсуляция – объединение в единое целое данных и алгоритмов обработки этих данных. В рамках ООП данные называются переменными или полями объекта, а алгоритмы – функциями или методами объекта.

ОБЩИЕ СВЕДЕНИЯ

Наследование – свойство объектов порождать своих потомков. Объект-потомок автоматически наследует от родителей все поля и методы, может дополнять объекты новыми полями и заменять (перекрывать) методы родителя или дополнять их.

ОБЩИЕ СВЕДЕНИЯ

Полиморфизм – свойство родственных объектов (т.е. объектов, имеющих одного общего родителя) решать схожие по смыслу проблемы разными способами.

на сегодняшний день ООП является одной из ведущих методологий в современных языках программирования.

ОБЩИЕ СВЕДЕНИЯ

Язык C++ относится к классу универсальных языков, поскольку с его помощью можно решить очень широкий круг задач, выполняемых на ЭВМ.

Среди современных алгоритмических языков язык C++ является, пожалуй, одним из самых популярных и распространенных, но наиболее эффективно его применение в написании системных программ-трансляторов, операционных систем, экранных интерфейсов, в обслуживании инструментальных средств.

В большинстве случаев программы, выполненные на языке C++, по быстродействию сравнимы с программами, написанными на Ассемблере.

ОБЩИЕ СВЕДЕНИЯ

Основные особенности языка C++ следующие:

- в нем реализованы некоторые операции низкого уровня;
- его базовые типы данных совпадают с типами данных языка Ассемблера;
- несмотря на присутствие таких составных объектов, как массивы и структуры, язык не допускает обращения с ними как с единым циклом;
- широко использует указатели на переменные и функции;
- удобным средством для передачи параметров являются ссылки;
- считается языком для профессионалов, поэтому многое «доверяет» программисту: даже на такие важные действия, как преобразование типов, налагаются лишь незначительные ограничения;
- несмотря на широкие возможности, невелик по объему за счет того, что практически все выполняемые функции оформлены в виде подключаемых библиотек.

ОБЩИЕ СВЕДЕНИЯ

Программа на C++ состоит из объявлений (переменных, констант, типов, классов, функций) и описаний функций.

Среди функций всегда имеется главная - main для консольных приложений (работающих с WIN32) или WinMain для приложений Windows. Именно эта главная функция выполняется после начала работы программы. Обычно в C++Builder эта функция очень короткая и выполняет только некоторые подготовительные операции, необходимые для начала работы. А далее при объектно-ориентированном подходе работа приложения определяется происходящими событиями и реакцией на них объектов.

Как правило, программы строятся по модульному принципу и состоят из множества модулей. Принцип модульности очень важен для создания надежных и относительно легко модифицируемых и сопровождаемых приложений. Четкое соблюдение принципов модульности в сочетании с принципом скрытия информации позволяет внутри любого модуля проводить какие-то модификации, не затрагивая при этом остальных модулей и головной файл.

ОБЩИЕ СВЕДЕНИЯ

В C++Builder все объекты компонентов размещаются в объектах - формах. Для каждой формы, которую вы проектируете в своем приложении, C++Builder создает отдельный модуль. Именно в модулях и осуществляется программирование задачи. В обработчиках событий объектов - форм и компонентов, вы помещаете все свои алгоритмы. В основном они сводятся к обработке информации, содержащейся в свойствах одних объектов, и задании по результатам обработки свойств других объектов. При этом вы постоянно обращаетесь к методам различных объектов.

Согласно принципам скрытия информации обычно текст модуля разделяют на заголовочный файл интерфейса, который содержит объявления классов, функций, переменных и т.п., и файл реализации, в котором содержится описание функций. Стандартное расширение файлов реализации - .cpp. Стандартное расширение заголовочных файлов - .h.

После того, как программа написана, на ее основе должен быть создан выполняемый файл (модуль). Этот процесс осуществляется в несколько этапов.

ОБЩИЕ СВЕДЕНИЯ

1. Сначала работает препроцессор, который преобразует исходный текст. Препроцессор осуществляет преобразования в соответствии со специальными директивами препроцессора, которые размещаются в исходном тексте. Препроцессор может в соответствии с этими директивами включать тексты одних файлов в тексты других, развертывать макросы - сокращенные обозначения различных выражений и выполнять множество других преобразований.

2. После окончания работы препроцессора начинает работать компилятор. Его задача - перевести тексты модулей в машинный (объектный) код. В результате для каждого исходного файла .cpp создается объектный файл, имеющий расширение .obj.

3. После окончания работы компилятора работает компоновщик, который объединяет объектные файлы в единый загрузочный выполняемый модуль, имеющий расширение .exe. Этот модуль можно запускать на выполнение.