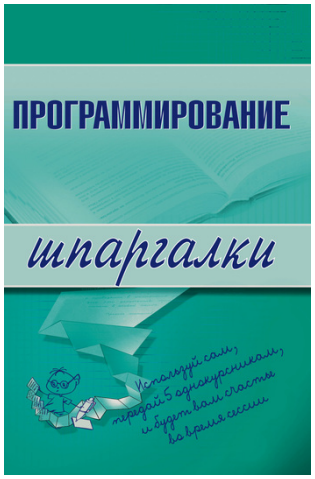
ПРОГРАММИРОВАНИЕ
шпаргалки

Ирина Сергеевна Козлова

Программирование

Серия «Шпаргалки»

Текст предоставлен правообладателем
http://www.litres.ru/pages/biblio_book/?art=180297
Программирование: ЭКСМО; Москва; 2008
ISBN 978-5-699-26658-6

Аннотация

Информативные ответы на все вопросы курса «Программирование» в соответствии с Государственным образовательным стандартом.

Содержание

Ирина Сергеевна Козлова
Программирование

1. Системы программирования

Машинно-ориентированные языки являются машинно-зависимым языком программирования. Основные конструктивные средства подобных языков дают возможность учитывать особенности архитектуры и принципов работы каждой ЭВМ.

Они позволяют записывать программу в виде, допускающем ее реализацию на ЭВМ с различными типами машинных операций, привязка к которым осуществляется соответствующим транслятором.

Язык СИ обладает некоторыми особенностями:

1) максимально используются возможности определенной вычислительной архитектуры; из-за этого программы на языке СИ компактны и работают эффективно;

2) обладает максимальными возможностями использования огромных выразительных возможностей современных языков высокого уровня. Процедурно-ориентированные языки чаще всего применяются для описания алгоритмов решения широкого класса задач; среди таких языков – Фортран, Кобол, Бейсик, Паскаль.

Проблемно-ориентированные языки применяются при описании процессов обработки информации в более узкой, специфической области; чаще всего применяются языки: РПГ, Лисп, АПЛ, GPSS.

Объектно-ориентированные языки программирования применяют в случае разработки программных приложений для широкого круга различных задач, которые имеют общность в реализуемых компонентах.

Интерпретация – пооператорная трансляция и последующее выполнение оттранслированного оператора исходной программы. Существуют следующие основные недостатки метода интерпретации:

1) интерпретирующая программа должна находиться в памяти ЭВМ в течение всего процесса осуществления исходной программы. То есть она должна занимать некоторый определенный объем памяти;

2) процесс трансляции одного и того же оператора повторяется столько раз, сколько должна исполнять эта команда в программе. Это является причиной резкого снижения производительности работы программы.

Но трансляторы-интерпретаторы широко распространены, так как они поддерживают диалоговый режим.

Процессы трансляции и выполнения при компиляции делятся во времени: первоначально исходная программа в полном объеме переводится на машинный язык, потом оттранслированная программа может многократно исполняться. Для трансляции методом компиляции нужен неоднократный «просмотр» транслируемой программы, т. е. трансляторы-компиляторы многопроходны. Трансляция методом компиляции именуется объектными модулями. Это эквивалентная программа в машинных кодах. Нужно, чтобы перед исполнением объектный модуль обработался особой программой операционной системы и преобразовался в загрузочный модуль.

Применяют кроме этого трансляторы интерпретаторы-компиляторы, которые объединяют в себе достоинства обоих принципов трансляции.

2. Классификация языков программирования высокого уровня

Высокоуровневые языки программирования применяются в машинно-независимых системах программирования. Подобные системы программирования в сравнении с машинно-ориентированными системами более просты в применении.

Языки программирования высокого уровня делятся на определенные группы:

1) процедурно-ориентированные языки, которые употребляются для записи процедур или алгоритмов обработки информации на любом круге задач:

а) язык Фортран (Fortran) (от Formulae Translation – «преобразование формул»). Фортран является одним из старейших языков программирования высокого уровня. Его существование и применение объясняется простотой его структуры;

б) язык Бейсик (Basic), который можно расшифровать как «Beginner's All-purpose Symbolic Instruction Code» (BASIC) – «многоцелевой символический обучающий код для начинающих», применяется с 1964 г. как язык для обучения программированию;

в) язык СИ (C), используется с 1970-х гг. как язык системного программирования специально для написания операционной системы UNIX. В 1980-е гг на основании языка С разработали язык С++, который включает в себя язык С и дополнен средствами объектно-ориентированного программирования;

г) язык Паскаль (Pascal) получил свое название в честь французского ученого Б. Паскаля. Его начал применять с 1968–1971 гг. Н. Вирт. При создании Паскаль использовали для обучения программированию, но впоследствии он стал применяться для разработки программных средств в профессиональном программировании; 2) проблемно-ориентированные языки применяются для разрешения целых классов новых задач, которые появляются при постоянном расширении области применения вычислительной техники:

а) язык Лисп (Lisp – List Information Symbol Processing) изобрел в 1962 г. Дж. Маккарти. Изначально он использовался как средство работы со строками символов. Лисп применялся в экспертных системах, системах аналитических вычислений и т. п.;

б) язык Пролог (Prolog – Programming in Logic) предназначается для логического программирования в системах искусственного интеллекта;

3) объектно-ориентированные языки, которые развиваются и в наше время. Большинство из таких языков – развитые версии процедурных и проблемных языков, но программирование с помощью языков такой группы более наглядно и просто. Среди таких языков можно выделить следующие:

а) Visual Basic (Basic);

б) Delphi (Pascal);

в) Visual Fortran (Fortran);

г) C++ (C);

д) Prolog++ (Prolog).

3. Переменные Visual Basic

В Visual Basic переменные накапливают информацию (значения). При их применении Visual Basic занимают область в памяти компьютера, которая предназначена для сохранения этой информации. Имена переменных, составленные из символов, могут иметь длину в 255 символов. Они начинаются с буквы, затем могут находиться другие буквы, цифры или символы подчеркивания. Регистр символов и наименований переменной не важен.

Все символы в имени переменной значимы, но их регистр не имеет значения. BASE обозначает такую же переменную, что и base. Но Base, Base 1 и Bas*e I являются различными переменными*. Visual Basic всегда заменяет первую букву переменной заглавной при определении.

Применение осмысленных имен помогает документировать текст программы и позволяет сделать процесс ее отладки намного легче. Выразительное имя переменной служит прекрасным способом объяснения смысла применения многих инструкций в коде программы.

Именем новых переменных не могут быть зарезервированные слова; например, Print не подходит для этого. Но такие слова могут использоваться как часть имени переменной, например: Printlt. Visual Basic будет показывать сообщение об ошибке, когда программист использует зарезервированное слово как название своей переменной, причем обычно непосредственно после нажатия клавиши ENTER.

Одно из наиболее общих соглашений об именах переменных состоит в использовании заглавных букв в начале каждого из слов, составляющих данное имя (например, Printlt, а не Printit). Данное соглашение называется «имена переменных со смешанным регистром». Иногда применяется и символ подчеркивания (например, PrintIt*)*, но его применяют не часто, так как это отнимает много места и иногда вызывает проблемы при отладке.

Visual Basic способен работать с 14 стандартными типами переменных. Также можно определить собственный тип данных. Рассмотрим некоторые из них, которые в основном применяются при работе с данными. String

Строковые переменные предназначены для того, чтобы хранить символы. Обозначить такой тип можно несколькими способами. Например, обозначать данный тип переменной с помощью добавления символа «$» к концу ее имени, например: AStringVariable$. Теоретически данная переменная может иметь до нескольких миллиардов символов. Однако на компьютере данное число будет намного меньше, так как накладываются ограничения на объемы оперативной памяти, ресурсы Windows или число символов, используемых в форме.

Наиболее часто строковые переменные применяются для выбора из полей ввода. К примеру, если есть поле ввода с именем Text1, в этом случае оператор ContentOfText1S = Text1.Text присваивает строку из поля ввода переменной в левой части такого оператора.

4. Типы переменных

Integer

Целочисленные переменные способны хранить только не очень большие целые числа, которые располагаются в диапазоне от –32768 до +32767. Арифметические операции над подобными числами производятся очень быстро. Для обозначения подобного типа применяется символ «%».

Long Integer

Подобный тип впервые был применен в языке QuickBASIC. В этих переменных располагаются целые значения от –2 147 483 648 до +2 147 483 647. Обозначается символом «&». Арифметические действия над приведенными числами выполняются тоже очень быстро, и в случае работы с процессором 386DX или 486DX обнаруживается только небольшая разница в скорости вычислений между Long Integer и Integer.

Single Precision

Идентификатором для таких чисел является символ «!». Такой тип переменной дает возможность хранить дробные числа, точность которых до седьмой цифры. То есть если получается результат 12345678.97, то часть 8.97 не точна. Результат может иметь значение, к примеру, 12345670.01. Длина чисел может иметь 38 знаков. Произведения математических операций с данными переменными тоже будут приблизительными. Кроме того, арифметические действия производятся медленнее, чем с целочисленными переменными.

Double Precision

Переменные подобного типа дают возможность хранить числа с точностью до 16 цифр и длиной до 300 4б символов. Идентификатором служит «#». Вычисления с ними тоже приблизительны, а скорость их не очень большая. Чаще всего переменные типа Double Precision применяются для научных расчетов. Currency

Этого типа не существовало в версиях GW-BASIC и QuickBASIC. Его применяют для того, чтобы не допускать ошибок при преобразовании десятичных чисел в двоичную форму и наоборот. Такой тип может иметь до 4 цифр после запятой и до 14 – перед ней. Внутри этого диапазона вычисления являются точными. Идентификатор такой переменной – символ «@». Так как все арифметические операции, кроме сложения и вычитания, производятся так же медленно, как и в случае переменных с двойной точностью, такой тип более предпочтителен для проведения финансовых расчетов.

Date

С помощью такого типа данных можно хранить значения времени и даты в промежутке от полуночи 1 января 100 года до полуночи 31 декабря 9999 года. Подобные значения в тексте программ обозначенысимволами «#», например: Millenium = #January 1, 2000#.

При введении только значения даты Visual Basic полагает, что время соответствует 00:00.

5. Целочисленные, переменного типа и переменные данных

Byte

Байтовый тип нов в Visual Basic и используется для хранения целых чисел от 0 до 255. Его применение дает возможность значительно экономить оперативную память и сократить размер массивов по сравнению с предыдущими версиями Visual Basic. К тому же его применяют при работе с двоичными файлами.

Boolean

Булев тип данных способен хранить только два значения: True или False. Его применение вместо целочисленных переменных представляет собой хороший стиль программирования.

Variant

Такой тип был введен в Visual Basic 5 из версии 2.0. Переменная типа variant способна содержать данные любого типа. Если Visual Basic не распознает тип принимаемых данных, следует использовать variant.

Тип информации не имеет значения, так как variant способен содержать любой тип данных (численный, дата и время, строковый). Visual Basic автоматически совершает необходимые преобразования данных, т. е. не стоит беспокоиться об этом. Однако можно применять встроенные функции для проверки типа данных, которые хранятся в переменной типа variant. С их помощью можно легко проверить, правильно ли пользователь вводит информацию.

Применение variant делает работу программы более медленной, так как необходимо время и ресурсы для того, чтобы произошло преобразование типов. К тому же многие программисты понимают, что применение автоматических преобразований типов данных является причиной неаккуратного вида программ. Причина использования variant заключается в возможных ошибках при преобразовании типов непосредственно.

В отличие от множеств других версий BASIC, в программе Visual Basic нельзя применять имена переменных, которые отличаются только типом (идентификатором), например A% и A!. В случае попытки применения двойного имени возникает ошибка «двойное определение» (duplicate difmition), когда происходит запуск программы.

При первом применении переменной Visual Basic временно присваивает ей пустое значение и тип variant. Такое значение пропадает в тот момент, когда переменной присваивают реальное имя. Любой тип данных имеет свое «пустое» значение. В случае строковых переменных это строка нулевой длины («»). В случае численных переменных – ноль. Полагаться следует только на значения по умолчанию, если они являются документированными (например, в комментариях). В противном случае появится множество трудно уловимых ошибок. Исходя из этого, следует инициализировать величины переменных в первых строках процедур обработки событий.

Одной из самых распространенных задач является обмен значениями между двумя переменными. Важно отметить, что разработчики Visual Basic убрали из языка оператор Swap, применяемый в QuickBASIC. Поэтому код следует писать самим.

6. Объявление переменных: оператор Dim для различных типов данных

Чаще всего люди стараются не пользоваться идентификаторами при обозначении типа переменной (тем более для таких типов, как дата/время). Вместо этого они применяют оператор Dim. Подобная операция называется объявлением. Объявление типов переменных при осуществлении обработки событий перед их использованием – естественно, с комментариями – представляет собой хороший стиль в программировании. Это также дает возможность улучшить «читабельность» текстов программ.

Если переменную объявили с помощью оператора Dim, в случае применения переменной с тем же именем и другим идентификатором типа будет наблюдаться ошибка «двойное определение» при запуске программы. К примеру, если следующее выражение Dim Count As Integer объявляет переменную Count, то нельзя применять переменные Counts, Count! Count# и Count®. Следует использовать только имя Count%, но это всего лишь другая форма для имени переменной Count.

Чтобы присвоить переменной тип variant, используют оператор Dim без As:

Dim F00 считает F00 переменной типа variant.

Можно написать и следующим образом: Dim Foo As Variant – это проще для прочтения.

Каждая информация, которая должна быть доступна всем процедурам обработки событий, относящихся к форме, размещается в разделе (General) данной формы.

Для размещения Option Explicit в раздел (General), следует выполнить следующие действия.

1. Открыть окно Code.
2. Выбрать объект (General) из списка объектов, которые предлагаются в окне Object.
3. Выбрать (Declaration) из списка Proc.
4. Ввести Option Explicit.

Часто применять объявления на уровне формы в разделе (General) необходимо, когда пользователь экспериментирует с примерами программ из справочной системы.

Для того чтобы копировать пример программы из справочной системы, следует использовать кнопку Copy в окне Code для примера. После этого можно использовать пункт Paste из меню Edit для помещения примера в окно Code. Когда Visual Basic встречает команду Option Explicit, он перестает позволять использовать необъявленные переменные. Если попробовать все же использовать такую переменную, будет показано сообщение об ошибке.

Для применения обязательного объявления типа переменной можно пользоваться страницей Editor диалоговой панели Tools|0ptions. Программист всегда устанавливает такой флажок. После этого оператор Option Explicit автоматически помещается в те места кода, где это необходимо.

7. Изменение значений по умолчанию для типов, область видимости

Пусть в новой программе почти все переменные являются целочисленными. Тогда удобно осуществлять их объявление так, чтобы переменная, для которой не указан тип, больше не объявлялась как variant. Для этого применяется оператор DefType.

Например, определить соглашение, что все переменные, которые начинаются с I, будут целочисленными, с помощью Deflnt I. Затем оператор Dim I будет объявлять переменные типа integer. Основные типы разных операторов DefType, которые чаще всего используются:

Deflnt диапазон букв (для integer);
DefLng диапазон букв (для long integer);
DefSng диапазон букв (для single precision);
DefDbl диапазон букв (для double precision);
DefCur диапазон букв (для currency);
DefStr диапазон букв (для string);
DefVar диапазон букв (для variant);
DefBooi диапазон букв (для boolean);
DefByte диапазон букв (для byte);
DefDate диапазон букв (для date).

Не всегда здесь применяются заглавные буквы: DefStr s-Z и Def-Str S-Z функционируют одинаково. Соглашения о формах можно всег-да изменить, применяя идентификатор или оператор Dim для каждой из переменных. Оператор DefType располагают в том же разделе (General), что и Option Explicit.

При программировании используют термин «область видимости», если хотят сказать о возможности применения переменной из одной части программы в другой ее части. В старых языках все переменные были применимы во всех частях программы, поэтому сохранение целостности наименований было огромной проблемой. Например, если в приложении применялись две переменные Total, то их значения уничтожали друг друга.

Решение подобной проблемы в современных языках высокого уровня, среди которых и Visual Basic, заключается в изолировании переменных внутри процедур. Пока это не определено специальным образом, значение переменной Total в одной процедуре никак не влияет на переменную с таким же именем в другой процедуре. На языке программирования такой подход означает, что данные переменные локальны по отношению к процедурам, если не указано иначе. Например, процедура обработки события чаще всего не имеет никакой связи с другой процедурой того же типа. Обычно в работе не используют переменные по умолчанию. Если нужно быть уверенным, что эта переменная является локальной, нужно объявить ее внутри процедуры события, применяя при этом оператор Dim.

8. Используемые символы языка СИ

Практически все символы, которые применяются в языке СИ, можно разделить на пять групп:

1. Символы, которые применяются для образования ключевых слов и идентификаторов. К ним относят прописные и строчные буквы английского алфавита и символ подчеркивания. Важно отметить, что одинаковые прописные и строчные буквы являются различными символами, так как обладают различными кодами.

2. Прописные и строчные буквы русского алфавита и арабские цифры.

3. Символы, образующие нумерацию, и специальные символы. Данные символы применяются для организации процесса вычислений, а также для передачи компилятору некоторого набора инструкций.

4. Управляющие и разделительные символы. К данной группе относят: пробел, символы табуляции, перевода строки, возврата каретки, новая страница и новая строка. Подобные символы призваны отделять друг от друга объекты, определяемые пользователем, среди которых константы и идентификаторы. Ряд разделительных символов считается компилятором как один символ (последовательность пробелов).

5. Кроме представленных групп символов в языке СИ широко применяются так называемые управляющие последовательности. Это специальные символьные комбинации, которые используются в функциях ввода и вывода информации. Управляющая последовательность начинается с обратной дробной черты (\) (обязательный первый символ) и комбинации латинских букв и цифр.

Последовательности вида \ddd и \xddd (d является цифрой) дают возможность представить символ из кода ПЭВМ как ряд восьмеричных или шестнадцатеричных цифр соответственно. К примеру, символ возврата каретки можно записать различными способами:

1) \r – общая управляющая последовательность;

2) \015 – восьмеричная управляющая последовательность;

3) \x00D – шестнадцатеричная управляющая последовательность.

Важно отметить, что в строковых константах следует задавать все три цифры в управляющей последовательности. Так, отдельную управляющую последовательность \n (переход на новую строку) можно переписать так: \010 или \xA. Однако в строковых константах нужно задавать все три цифры, иначе символ или символы, идущие за управляющей последовательностью, будут считаться ее недостающей частью. К примеру: «ABCDE\x009FGH». Эта строковая команда будет напечатана с применением определенных функций языка СИ как два слова ABCDE FGH, между которыми располагаются 8 пробелов. Если указать неполную управляющую строку «ABCDE\x09FGH», то на печати появится ABCDE=|=GH, вследствие того, что компилятор воспримет последовательность \x09F как символ «=+=». Важно отметить, что если обратная дробная черта предшествует символу, не являющемуся управляющей последовательностью (т. е. не включенному в табл.

4) и не являющемуся цифрой, то данная черта не учитывается, а символ представляется как литеральный.

9. Константы языка СИ

Константы – это перечисление величин в программе. В языке СИ можно выделить четыре вида констант: целые константы, константы с плавающей запятой, символьные константы и строковые литералы.

Целая константа – это десятичное, восьмеричное или шестнадцатеричное число, представляющее целую величину в одной из известных форм: десятичной, восьмеричной или шестнадцатеричной. Десятичная константа включает в себя одну или несколько десятичных цифр, при этом первая цифра не должна быть нулем (иначе число будет воспринято как восьмеричное). Восьмеричная константа включает в себя обязательный нуль и одну или несколько восьмеричных цифр (среди цифр не должно быть восьмерки и девятки, так как данные цифры не входят в восьмеричную систему счисления). Шестнадцатеричная константа начинается с непременной последовательности 0x или 0X и включает в себя одну или несколько шестнадцатеричных цифр, которые являются набором цифр шестнадцатеричной системы счисления: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Если необходимо образовать отрицательную целую константу, то применяется знак «-» перед записью константы (который называется унарным минусом). К примеру: –0x3A, –098, –36. Любой целой константе присваивается тип, который определяет преобразования, которые должны быть выполнены, если константа применяется в выражениях. Тип константы определяется так:

1) десятичные константы – это величины со знаком, и им присваивается тип int (целая) или long (длинная целая) по значению константы. Если константа меньше 32 768, то ее тип – int, иначе – long;

2) восьмеричным и шестнадцатеричным константам присваивается тип int, unsigned int (беззнаковая целая), long или unsigned long по величине константы.

Для того чтобы каждую целую константу определить типом long, в конце константы ставится буква «l» или «L». Пример: 2l, 9l, 138L, 0905L, OX2911L.

Константа с плавающей точкой – десятичное число, которое представлено в виде действительной величины с десятичной точкой или экспонентой. Формат записывается так: [цифры].[цифры] [E|e [+|-] цифры].

Число с плавающей точкой включает себя целую и дробную части и (или) экспоненты. Константы с плавающей точкой – это положительные величины удвоенной точности, тип которых double. Для того чтобы определить отрицательную величину, следует сформировать константное выражение, которое состоит из знака минуса и положительной константы.

10. Примеры использования констант языка СИ

119.75, 1.7E2, –0.825, 0.035, –0.89E2.

Символьная константа является символом, который заключен в апострофы. Управляющая последовательность является одиночным символом, допустимо ее применять в символьных константах. Значением символьной константы считается числовой код символа.

Примеры:

' ' – пробел;

'\\ – обратная дробная черта;

'W' – буква W;

'\n' – символ новой строки;

'\v' – вертикальная табуляция.

Символьные константы принадлежат типу int, и при преобразовании типов к ним приписывается соответствующий знак.

Строковая константа (литерал) представляет собой последовательность символов, среди которых строковые и прописные буквы русского и латинского алфавита или цифры, которые заключены в кавычки («»). Например: «Школа N 35», «город Саратов», «YZPT КОД». Важно заметить, что все управляющие символы, кавычки («»), обратная дробная черта (\) и символ новой строки в строковом литерале и в символьной константе являются соответствующими управляющими последовательностями. Любая управляющая последовательность записывается как один символ. К примеру, при печати фразы «Школа \n N 35» часть «Школа» будет располагаться на одной строке, а часть «N 35» – на следующей. Знаки строкового выражения 10б сохраняются в области оперативной памяти. В конец любого строкового литерала компилятор прибавляет нулевой символ, который представляется управляющей последовательностью \0. Строковое выражение имеет тип char []. То есть строка представляется как массив символов. Количество элементов массива соответствует числу символов в строке плюс 1, вследствие того, что нулевой символ (символ конца строки) также служит элементом массива. Все строковые выражения рассматриваются компилятором как различные объекты. Строковые литералы способны находиться на нескольких строках. Такие фразы формируются на основе применения обратной дробной черты и клавиши «ввод». Обратная черта и символ новой строки рассматривается компилятором, в результате чего следующая строка служит продолжением предыдущей. К примеру, «строка неопределенной \n длины» полностью аналогична фразе «строка неопределенной длины». С целью сцепления строковых фраз можно применять символ (или символы) пробела. Когда в программе присутствуют два или более строковых выражения, которые разделены только пробелами, они будут рассматриваться как одна символьная строка. Данный принцип можно применять для формирования строковых литералов, занимающих более одной строки.

11. Идентификатор. Ключевые слова

Идентификатором называется последовательность цифр, букв и специальных символов. При этом первой стоит буква или специальный символ. Для получения идентификаторов можно использовать строчные или прописные буквы латинского алфавита. Специальным символом может служить символ подчеркивания *()*. Два идентификатора, для получения которых применяются совпадающие строчные и прописные буквы, считают различными. К примеру: abc, ABC, A328B, a328b. Компилятор допускает всякое количество символов в идентификаторе, но значим только первый 31 символ. Идентификатор образуется на этапе объявления переменной, функции, структуры и т. п. После этого его можно применять в последующих операторах разрабатываемой программы. Важно отметить некоторые особенности при выборе идентификатора. Во-первых, идентификатор и ключевое слово не должны совпадать. Также не должно быть совпадения с зарезервированными словами и названиями функций библиотеки компилятора языка СИ.

Во-вторых, важно обратить особое внимание на применение символа подчеркивания *() первым символом идентификатора*, так как идентификаторы выстраиваются так, что, с одной стороны, могут совпадать с именами системных функций и (или) переменных, но при этом при применении таких идентификаторов программы могут стать непереносимыми, т. е. их нельзя применять на компьютерах других типов.

В-третьих, на идентификаторы, применяемые для определения внешних переменных, должны быть наложены ограничения, которые формируются используемым редактором связей. Кроме того, использование различных версий редактора связей или различных редакторов определяет различные требования на имена внешних переменных.

Ключевыми словами называются зарезервированные идентификаторы, наделенные определенным смыслом. Их можно применять только в соответствии со значением, известным компилятору языка СИ.

Приведем список ключевых слов:

auto double int struct break else long switch register tupedef char extern return void case float unsigned default for signed union do if sizeof volatile continue enum short while.

При этом в определенных версиях реализации языка СИ зарезервированными словами являются следующие: *asm*, fortran, near, far, cdecl, huge, pascal, interrupt.

Ключевые слова far, huge, near дают возможность определить размеры указателей на области памяти. Ключевые слова *asm*, cdecl, fortran, pascal используются для организации связи с функциями, которые написаны на других языках, а также для применения команд языка ассемблера непосредственно в теле будущей программы на языке СИ. Ключевые слова не могут применяться в качестве идентификаторов.

12. Комментарии. Исходные файлы

Комментарием является набор символов, игнорируемых компилятором. Но на данный набор символов накладываются определенные ограничения. Внутри набора символов, представляющих комментарий, не может быть специальных символов, которые определяют начало и конец комментариев, соответственно (/* и */). Важно показать, что комментарии способны заменить одну или несколько строк.

Приведем конкретные примеры:

/* комментарии к программе */

/* начало алгоритма */

или /*

Комментарии могут быть записаны в любом виде, но следует быть осторожным и не допустить внутри последовательности, которая игнорируется компилятором, появления оператора программы, который также будет игнорироваться */. При этом производится неправильное определение комментариев.

/* комментарии к алгоритму /* решение краевой задачи */ */

или

/* комментарии к алгоритму решения */ краевой задачи */

Обычная СИ-программа является определением функции main, которая для выполнения определенных действий вызывает другие функции. Связь между функциями производилась по данным через передачу параметров и возврата значений функций. Однако компилятор языка СИ дает возможность также разбить программу на несколько отдельных частей, которые являются исходными файлами, оттранслировать любую часть отдельно и после этого объединить все части в один выполняемый файл при помощи редактора связей. При данной структуре исходной программы функции, располагающиеся в разных исходных файлах, могут применять глобальные внешние переменные. Все функции в языке СИ по определению внешние и постоянно доступны из каждого файла. Для выполнения определяемой функцией каких-либо действий она должна применять переменные. В языке СИ все переменные объявляются до их применения. Объявления определяют соответствие имени и атрибутов переменной, функции или типа. Определение переменной приводит к выделению памяти для хранения ее значения. Класс отводимой памяти определяется спецификатором вида памяти и задает время жизни и область видимости переменной, которые связаны с понятием блока программы. В СИ блоком является ряд объявлений, определений и операторов, располагающихся в фигурных скобках.

Можно выделить два вида блоков – составной оператор и определение функции, которые состоят из составного оператора (тела функции) и заголовка функции, который находится перед телом функции (в него входят имя функции, типы возвращаемого значения и формальных параметров). Блоки могут состоять из операторов, но не определения функций. Внутренний блок носит название вложенного, а внешний – объемлющего.

Временем при жизни называется интервал времени выполнения программы, за который программный объект (переменная или функция) существует. Время жизни переменной бывает локальным или глобальным. Переменная с глобальным временем жизни обладает распределенной для нее памятью и определенным значением на протяжении всего времени выполнения программы.

Конец ознакомительного фрагмента.