Аурика Луковкина

Программирование. Шпаргалка

Аурика Луковкина

Программирование. Шпаргалка

Текст предоставлен правообладателем
http://www.litres.ru/pages/biblio_book/?art=9094553
Программирование. Шпаргалка: Научная книга; 2009

Аннотация

Настоящее издание поможет систематизировать полученные ранее знания, а также подготовиться к экзамену или зачету и успешно их сдать.

Содержание

Программирование. Шпаргалка

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, Пас-

каль.

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 от-

транслированная 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 (С), 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 развивают-

ся 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 Base_ 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 зарезервиро-

ванное 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, Print_It), 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 –32 768 до +32 767. 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 12 345 678.97, то часть 8.97 не точна. Результат может иметь значение, к примеру, 12 345 670.01. Длина чисел может иметь 38 знаков. Произведения математических операций с данными переменными тоже будут при-

близительными. 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 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 обозначены символами «#», 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 комментари-

ях). 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.

Конец ознакомительного фрагмента.