Методическое пособие

по программированию

на Quick Basic

**I Этапы решения задач на ЭВМ.**

1. Чёткая формулировка задачи, выделение выходных и входных данных.
2. Выбор метода решения.
3. Разработка алгоритма на основе выбранного метода.
4. Выбор структуры данных.
5. Запись разработанного алгоритма на языке программирования.
6. Отладка программы.
7. Анализ полученных результатов.

**II Алгоритм и блок-схема.**

*Алгоритм* – понятное и точное предписание, ведущее от исходных данных к получению результата.

***Формы представления алгоритмов*:**

Словесная запись. Неформализованная запись алгоритма на естественном языке, например, рецепт приготовления манной каши.

Блок-схема. Наглядное графическое изображение структуры алгоритма. Она строится из блоков, соединенных стрелками.

|  |  |  |  |
| --- | --- | --- | --- |
| http://festival.1september.ru/articles/412697/img1.JPG | начало и конец алгоритма; | http://festival.1september.ru/articles/412697/img2.JPG | Выбор направления выполня алгоритма в зависимости от некоторых переменных условий; |
| выполнение операции; |
| ввод и вывод данных.  |

Компьютерная программа. Жёстко формализованная запись алгоритма, ориентированная на исполнителя – ЭВМ. Для разработки компьютерных программ используются инструментальные средства, называемые языками программирования.

**III Структура программы на языке Quick-Basic**

Строка имеет следующий формат: [метка] оператор [: оператор] … [комментарий].

*Метка* - целое число от 0 до 65535 или последовательность букв и цифр, начинающаяся с буквы. Метка оканчивается двоеточием.

*Оператор* - команда для ЭВМ на выполнение определенного действия.

*Комментарий -* любой текст, расположенный после " ‘ " или "REM"

|  |  |
| --- | --- |
| Примеры: |  |
| START: | метка; |
| В=В+1 | оператор (присваивания); |
| А=А\*3: В=В+1 сложение | два оператора и комментарий. |

В строке не должно быть более 249 символов. Переменные в языке Quick-Basic могут обозначаться .любыми латинскими буквами и числами, но начинаться должны с буквы. Необходимо помнить, что Quick-Basic различает переменные по не более, чем восьми символам. Таким образом, переменные ABCDEFGHK и ABCDEFGHT являются одинаковыми. Пример записи переменных: А, АВ, G1, BLAK и т.д.

**IV Арифметические выражения.**

*Арифметическое выражение* состоит из чисел, переменных, функций (стандартные или определяемые в программе), знаков, скобок. Арифметические выражения записываются и вычисляются по правилам математики.

Примеры арифметических выражений.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Обычная запись | A+4 | 2A | AB/C | X/(S+D) | A2 |
| Запись на Basic | A+4 | 2\* A | A\* B/C | X(S+D) | A^2 |

**V Операторы ввода и вывода.**

**INPUT –** оператор**,** использующийся для ввода данных с клавиатуры во время выполнения программы. Общий вид оператора: INPUT vl [, v2[, …]], где vl, v2, ... - переменные. При выполнении этого оператора ЭВМ делает паузу, на экране высвечивается ? и ЭВМ ждёт ввода данных. На клавиатуре необходимо набрать числа, подлежащие вводу, в порядке, задаваемом списком ввода. Если чисел несколько, то они отделяются друг от друга запятыми. После этого нажимаем клавишу Enter, и эти значения присваиваются соответствующим переменным.

**PRINT** - оператор**,** использующийся для вывода информации на экран дисплея. Общий вид оператора: PRINT список [; список [; …]]. Список вывода может содержать имена переменных, числа, арифметические выражения и тексты. Текст заключается в кавычки и может содержать любые символы.

**INKEY$ -** оператор**,** использующийся для ввода символа нажатой клавиши. Общий вид оператора: I$=INKEY$, где I$ - переменная, которой присваивается символ. При выполнении этого оператора ЭВМ присваивает текстовой переменной символ нажатой в этот момент клавиши. Если клавиша не была нажата, то символьной переменной присвоится пустой символ (''''), и программа продолжит свою работу.

**DATA, READ, RESTORE** - эти операторы в программе всегда присутствуют одновременно. Оператор DATA имеет вид: DATA al [, a2[, …]], где al, a2, ... - константы. Оператор READ имеет вид: READ vl [, v2[, …]], где vl, v2, ... – вводимые переменные. Оператор DATA содержит данные, которые при выполнении операторов READ будут присваиваться переменным, перечисленным в списке ввода операторов READ, в порядке очереди. Данные из области DATA можно считывать повторно, используя оператор RESTORE. Общий вид: RESTORE [метка].

**VI Стандартные функции.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Запись на Basic | Математическое определение  | Запись на Basic | Математическое определение  | Запись на Basic | Математическое определение  |
| SIN(X) | Sin x | SQR(X) | http://festival.1september.ru/articles/412697/img3.JPG | INT(X) | целая часть x |
| COS(X) | Cos x | ABS(X) | |х| | SGN(X) | знак числа |
| RND(1) | датчик случайных чисел | A mod X | остаток от деления числа A на число X | FIX(X) | отбрасывание дробной части |

В качестве аргумента можно употреблять любое арифметическое выражение.

Рассмотрим, как работает датчик случайных чисел RND(1). Необходимо получить числа из промежутка [t, z]. Функция RND(1) даёт числа из промежутка от 0 до 1.

0 RND(1) 1 Умножим это неравенство на (z - t).

0 RND(1)\* (z - t) z – t Прибавим к этому неравенству t.

t RND(1)\* (z - t) + t z.

Таким образом, чтобы получить числа из промежутка [t, z], необходимо RND(1)\* (z - t) + t.

**VII Линейные алгоритмы.**

*Линейный алгоритм* – это набор команд, выполняемых последовательно во времени друг за другом.

Пример. Дана длина ребра куба. Найти объём данного куба.



*Программа.*REM
CLS
INPUT "введите ребро куба"; А
V = A^3
PRINT "объём куба равен"; V
END

**

***VIII. Разветвляющиеся алгоритмы.***

*Разветвляющийся алгоритм* – это алгоритм, содержащий хотя бы одно условие, в результате которого осуществляется переход на один из двух возможных шагов, называется разветвляющимся.

Блок-схема базовых структур:

|  |  |
| --- | --- |
| Развилка полная  | Развилка неполная |
| img7.JPG (8068 bytes) | img8.JPG (7138 bytes) |

В языке Basic для изменения последовательности выполнения операторов используются специальные операторы: безусловного перехода GОТО и условного перехода IF. Синтаксис для оператора безусловного перехода: GОТО метка. Синтаксис для оператора условного перехода: IF [условие] THEN [операторы\_1] ELSE [операторы\_2]. Условие содержит знаки сравнения, которые записываются следующим образом:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| = | < | > | <= | >= | <> |
| равно | меньше | больше | меньше или равно | больше или равно | не равно |

Условие может состоять из несколько условий, связанных логическими операциями. Логические операторы: NOT - не; AND - и; OR - или. Операторы\_1 выполняются, если условие истинно. Операторы\_2 выполняются, если условие ложно.

Пример. Нахождение значений функции у=|х-1|.

|  |  |
| --- | --- |
| *Блок-схема.* img9.JPG (13094 bytes) | *Программа.* Rem: Cls1:Input “введите значение переменной х”; XIF X-1<O THEN Y= - X+l ELSE Y=X-1PRINT “Y=”; Y: GOTO 1 |

**IX Циклические алгоритмы.**

*Циклический алгоритм* – это алгоритм, предусматривающий многократное повторение одного и того же действия над новыми данными.

*Арифметический цикл* – это цикл, в котором число повторений известно заранее или может быть вычислено.

В языке Basic существует специальный оператор для составления циклических алгоритмов. Это оператор цикла: FOR.

Синтаксис:

FOR K=K1 TО К2 [STEP S]

……………………………..

NEXT K

Переменная К - это управляющая переменная цикла.

Параметр цикла изменяется от начального значения К1 до конечного значения К2.

Величина S показывает, с каким шагом изменяется параметр цикла К.

По умолчанию S=1. NEXT

К - конец цикла.

Количество выполняемых циклов вычисляется по формуле: N =(K2-K1)/S+1.

Пример. Нахождение n!

Rem: Cls: P=1
Input “введите количество элементов”; N
FOR I=1 TO N
P=P\* I
NEXT
PRINT “n!=”; P
End

*Итерационный цикл* – это цикл, для которого нельзя указать число повторений, и проверка окончания которого происходит по достижению нужного условия.



Блок-схема базовых структур.

|  |  |
| --- | --- |
| Цикл - **пока** | Цикл - **до** |
| (может не выполняться ни разу) | (выполняется хотя бы раз) |
| http://festival.1september.ru/articles/412697/img11.JPG | http://festival.1september.ru/articles/412697/img12.JPG |

|  |  |
| --- | --- |
| Цикл - **пока** | Цикл - **до** |
| DO WHILE условие команды тела циклаLOOP | Тело цикла выполняется до тех пор, пока УСЛОВИЕ ИСТИННО. | DO команды тела циклаLOOP WHILE условие | Тело цикла выполняется до тех пор, пока УСЛОВИЕ ИСТИННО. |
| DO UNTIL условие команды тела циклаLOOP | Тело цикла выполняется до тех пор, пока УСЛОВИЕ ЛОЖНО. | DO команды тела циклаLOOP UNTIL условие | Тело цикла выполняется до тех пор, пока УСЛОВИЕ ЛОЖНО. |

*Сложный цикл* – это цикл, содержащий внутри себя один или несколько других циклов. Внутренний цикл должен быть целиком внутри внешнего цикла. Сложный цикл схематично можно представить так:

|  |  |
| --- | --- |
| img13.JPG (5081 bytes)  | Внешний цикл (1) содержит один внутренний (2). |
| Внешний цикл (1) содержит внутренний (2), который в свою очередь содержит внутренний цикл (3). |

|  |  |
| --- | --- |
| img14.JPG (4700 bytes) | Внешний цикл (1) содержит независимые внутренние циклы (2)...(n) |

**X Запись файла на диск. Считывание файла с диска.**

При работе в среде Quick-Basic на ПК IBM для записи на диск необходимо выполнить следующие действия:

1. Нажатием клавиши Alt выйти в верхнее меню.
2. Высветить слово File.
3. Нажатием Enter выйти в меню файлов.
4. Высветить слово Save ass....
5. Нажатием Enter выйти в меню записи.
6. Набрать имя программы и нажать Enter.

После чего: производится запись программы на диск под указанным именем с расширением bas.

Для считывания с диска производятся следующие действия:

1. Нажатием клавиши Alt выйти в верхнее меню.
2. Высветить слово File.
3. Нажатием Enter выйти в меню файлов.
4. Высветить слово Load ass....
5. Нажатием Enter выйти в меню записи.
6. Набрать имя программы и нажать Enter.

После чего производится считывание программы с диска под указанным именем.

**XI Массивы.**

*Массив* – это упорядоченный набор величин одного типа, обозначаемых одним именем. Каждый элемент массива имеет свой порядковый номер - индекс. Идентификатор элемента массива определяется именем массива и индексом. Минимальное значение индекса массива в Quick-Basic равно 0. Например: A(4), D$(2) и т.д.

Массивы бывают одномерные и многомерные. Массив в языке Quick-Basic определяется оператором DIM (DIMENSION - размер). Синтаксис: DIM имя (размер [, размер])

Пример. Вычислить сумму элементов массива D размерности (m, n). Элементы массива лежат в промежутке [-30;60].

Rem
Cls
S=0
Input “введите количество строк”; M
Input “введите количество столбцов”; N
DIM D (m, n)
FOR I=1 TO M
FOR J=1 TO N
D (I, J) = INT (RND(1)\* 90+30)
S = S + D (I, J)
PRINT D (I, J);
NEXT
NEXT
PRINT “”
PRINT “S =”; S
End

**XII Функции пользователя.**

При решении некоторых задач возникает необходимость использовать фунции, которых нет в памяти ЭВМ. В таких случаях в Quick-Basic есть возможность создавать свои функции (функции пользователя). Для того, чтобы использовать эти функции, необходимо их определить. Для этой цели используется оператор: DEF FN. Результатом выполнения функции является всегда единственное значение.

Синтаксис:

Для определения функций:

DEF FNимя (переменная [переменная, [, …]]), где переменная - формальный параметр.

Для использования функций:

FNимя (переменная [переменная, [, …]]), где переменная - фактический параметр.

Пример. Решить уравнение 2x2 + x - 5 методом дихотомии, где x0, x1 - концы отрезка в котором находится один корень, Е - погрешность.

Rem
Cls
DEF FNY(X)=2\* X^2+X-5
INPUT “введите первый конец отрезка”; X0
INPUT “ введите второй конец отрезка”; X1
INPUT “ введите погрешность”; E
1: Y0=FNY(X0) : Y1=FNY(X1)
X=X1-Y1\* (X0-X1)/(Y0-Y1)
Y=FNY(X)
IF Y\* Y0>0 THEN X0=X ELSE X1=X
IF ABS(X-X\* E)>E THEN GOTO 1
PRINT "X="; X
END .

**XIII Подпрограммы.**

Во многих языках программирования используются подпрограммы. Во-первых, это даёт, возможность записать повторяющуюся часть алгоритма только один раз, а, затем использовать её в нужном месте программы. Во-вторых, это позволяет сложную программу сделать более понятной и удобной.

Подпрограммы очень похожи на программы. Каждая подпрограмма представляет собой независимый алгоритм решения определенной небольшой задачи. Обычно подпрограмма имеет один или несколько входных параметров, для которых нужно решить данную задачу. После выполнения подпрограммы получается определённый результат. Это может быть один или несколько выходных параметров или определённое действие. Подпрограмма отличается от программы тем, что она не может выполняться самостоятельно, подпрограммы обязательно должны вызываться (использоваться) в программах.

Для вызова подпрограммы в Basic используется оператор GOSUB (SUB - от слова SUBROUTINE). Оканчивается подпрограмма оператором RETURN.

Синтаксис: GOSUB N, где N – метка.

Пример:

Rem
Cls
А=3: В=7: GOSUB 1
А=7: В=2: GOSUB 1
А=1: В=5: GOSUB 1
END
1: REM ‘Подпрограмма
S=A+B: P=A\* B
PRINT "А="; А : "В="; В
PRINT "сумма S="; S : "произведение P="; P
RETURN

**XIV Работа со строковыми переменными.**

Ранее мы рассматривали числовые переменные, но нельзя забывать, что язык Basic позволяет обрабатывать и другой вид переменных - символьные. Имена символьных переменных заканчиваются знаком $. Символы - это любая строка, заключённая в кавычки. Символьные переменные можно присоединить друг к другу операцией сложения.

Например:

|  |  |
| --- | --- |
| "А" | символ. |
| "Саше 10 лет." | строка символов. |
| А$ | символьная переменная. |
| F$="Абстракция" | присваивание символьной переменной строки символов. |
| А$="Саше"+"10 лет" | в результате переменная А$ будет содержать "Саше 10 лет". |

Рассмотрим функции, которые позволяют работать с символьными переменными, на примере: A$="INFORMATIKA", F$="FO", G$="12", N=4, C=73.

|  |  |  |
| --- | --- | --- |
| **Оператор** | **Назначение** | **Пример** |
| MID$ (A$, N, K) | Выделяет из A$, начиная с позиции N, K символов. | "ORM" |
| LEFT$ (A$, N) | Выделяет из A$ N левых символов. | "INFO" |
| RIGHT$ (A$, K) | Выделяет из A$ K правых символов. | "IKA" |
| STRING$ (N, F$) | Повторяет N раз строку F$. | "FOFOFOFO" |
| SPACE$ (N) | Повторяет N раз символ пробел. | " " |
| LNSTR ([K,] A$, F$) | Определяет номер вхождения F$ в A$, начиная с позиции K. | 3 |
| LEN (A$) | Возвращает количество символов в A$. | 11 |
| CHR$ (C) | Возвращает символ, десятичный код которого в ASCII равен C. | "I" |
| ASC (A$) | Переводит символ A$ в десятичный код ASCII. | 73 |
| STR$ (K) | Выдаёт строку цифр, являющихся составляющими числа K. | "3" |
| VAL (G$) | Переводит строку цифр в соответствующее число. | 12 |

**XV Графика в языке Quick-Basic**

В языке Quick-Basic после включения компьютера устанавливается, по умолчанию, текстовый режим. Поэтому для работы с графическими операторами необходимо установить режим работы графического экрана. Режим работы экрана дисплея устанавливается оператором SCREEN.

Синтаксис: SCREEN N, где N - это номер экрана.

N: 0, 1 - символьный экран Yamaha,
2, 3 - графический, для MSX1,
2-8 - графический, для. МSХ2,
0, 7 - символьный экран ПК IBM,
1-6, 8-13 - графический экран ПК IBM.

В Quick-Basic есть дополнительные операторы:

VIEW - для выделения прямоугольной части экрана,

WINDOW - для задания на экране удобной для данной задачи декартовой системы координат.

Синтаксис:

VIEW (X1, Y1) - (X2, Y2),

WINDOW (XA, YA) - (XB, YB).

Пример. Выделение на экране прямоугольника и получение обычной системы координат, в которой X изменяется от -6.28 до +6.28, a Y от -1 до +1.

SCREEN 2

VIEW (300, l00) - (639, 199)

WINDOW (-6.28, +1) - (6.28, -1)

Рассмотрим графические операторы.

|  |  |
| --- | --- |
| CLS | Очищает экран. |
| COLOR C1, C2, C3 | Устанавливает цвет изображения, фона и бордюра соответственно. |
| PSET (Х, Y), С | Выводит точку на экран с координатами (Х, Y) цветом С. |
| PRESET (Х, Y) | Ставит на экран точку с координатами (Х, Y) цветом фона. |
| LINE (X1,Y1) - (X2,Y2), C | Рисует отрезок с концами в точках (X1,Y1) и (X2,Y2) цветом С. |
| LINE (X1,Y1) - (X2,Y2), C, B[F] | Рисует [закрашенный] прямоугольник цветом С, где (X1,Y1) и (X2,Y2) – координаты одной из диагоналей. |
| CIRCLE (X, Y), R, C | Рисует окружность с центром (X, Y) радиусом R и цветом С. |
| CIRCLE (X, Y), R, C, L1, L2, K | Рисует дугу эллипса, где L1 - начальный, L2 - конечный угол в радианах, а К-коэффициент сжатия. |
| PAINT (X,Y), C | Закраска замкнутой области экрана цветом С. |
| POINT (Х, Y) | Возвращает цвет точки с координатами (Х, Y). |

В Quick-Basic есть дополнительные операторы.

|  |  |
| --- | --- |
| SCREEN ,, N, K | Выбор страницы, где N – активная страница, K – видимая страница. |
| PCOPY N, K | Копирует N страницу на K. |
| PALETTE C, A | Изменение цвета C, A - атрибут цвета. |

Таблица номеров цветов для режима SCREEN 7.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | чёрный | **4** | красный | **8** | серый | **12** | светло- красный |
| **1** | голубой | **5** | сиреневый | **9** | светло- голубой | **13** | светло- сиреневый |
| **2** | зелёный | **6** | коричневый | **10** | светло- зелёный | **14** | жёлтый |
| **3** | бирюзовый | **7** | белый | **11** | светло- бирюзовый | **15** | ярко-белый |

Для распечатки текста на графическом экране во многих языках Basic, в том числе и в Quick-Basic, достаточно воспользоваться оператором PRINT. Для указания положения курсора на экране, от которого будет печататься текст, нужно воспользоваться оператором LOCATE. Он же устанавливает курсор и на текстовом экране.

Синтаксис: LOCATE y, x, где х - от 0 до 40 (80), у - от 0 до 24.

Пример.

SCREEN 2
LOCATE 10, 30
PRINT "Привет"
END

**XVI Движение изображения.**

В различных версиях языкое Basic движения изображения добиваются по-разному. Но в основе лежит принцип рисования, стирания и рисования со смещением. В Quick-Basic есть дополнительные операторы PUT, GET. GET переносит изображение с экрана в массив. PUT переносит с массива на экран.

Синтаксис:

GET (X1, Y1) - (X2, Y2), A, где X1,Y1 - правый верхний угол изображения, X2,Y2 - левый нижний угол изображения, А - имя массива.

PUT (X, Y), A, [ключ], где X,Y - правый верхний угол изображения, А - имя массива.

Ключи:

PSET - рисует образ;

PRESET - рисует образ в инвертированном виде;

OR - накладывает образ на рисунок;

AND - рисует совпадающие точки;

XOR - рисует ранее нарисованный образ с сохранением фоновых эффектов мультипликации.

Массив должен быть описан ранее. Количество элементов массива, в котором запоминается изображение можно вычислить по приближённой формуле: N=4+(a+1)\* b/4, где a – ширина, а b – высота прямоугольной области в точках.

Наилучший эффект мультипликации создается следующим образом. Сначала прямоугольный фрагмент экрана заполняется и выдается в ту же область с ключом XOR. При этом прежнее изображение стирается. Далее повторно выполняется оператор PUT, но со смещением.

Пример. Движение рисунка

REM
DIM A (110)
SCREEN 7
CIRCLE (10, 10), 10, 4
CIRCLE (7, 6), 1.2, 4
CIRCLE (13, 6), 1.2, 4
LINE (10, 10)-(10, 11), 4
LINE (9, 15)-(11, 15), 4
GET (0, 0)-(20, 20), A
CLS
FOR i = 1 TO 5
x = INT (RND(1) \* 280)
y = INT (RND(1) \* 180)
PUT (x, y), A, XOR
FOR J = 1 TO 300000: NEXT
PUT (x, y), A, XOR
NEXT
END

**XVII Звуковое воспроизведение по нотам.**

Для звукового воспроизведения по нотам используется оператор PLAY.

Синтаксис: PLAY F$, где F$ - командная строка.

Команды:

|  |  |
| --- | --- |
| On | задаёт текущую октаву (0-6) |
| A – G | играет определённую ноту текущей октавы |
| # или + | диез  |
| - | бемоль |
| Ln | задает длительность каждой ноты (1-64) |
| Pn | задает паузу (1-64) |
| Tn | задает темп исполнения в четвертях в минуту (32-255)  |
| <или> | переходит на одну октаву вверх или вниз |
| Nn | играет определённую ноту (0-84) (0-сброс) |
| ML | вид исполнения legato |
| MN | вид исполнения normal |
| Ms | вид исполнения staccato |
| Mf | основное звучание |
| MB | фоновое звучание |

Пример.

1: PLAY "04AGFEFCO3BO4CDC3BO4CG4CDEFGA#AGAGFGF4GEGGEGEGA#AFAFAFAO

5CO4GEGEGEGA#AGFEFO5CO4AG"

GOTO 1