Ściąga – elementy języka programowania QBasic

QBasic jest programem **interpretacyjnym (interpreterem)**. Nie ma kompilatora. Może najnowsze wersje.
Program jest zintegrowany z edytorem tekstu.
Generalnie jest to interpreter. Interpretuje kolejne instrukcje programu źródłowego (z rozszerzeniem BAS i wykonuje).
Programy piszemy w czystym edytorze tekstowym (Edit, Notatnik, edytor Basica lub inny do pisania programów).

Przejście miedzy trybem programowym a bezpośrednim w QBasicu – klawisz **F6**.
W trybie bezpośrednim wykonuje się polecenia, głównie obliczenia bezpośrednio, np. 2 razy 3 pisze się:
? 2*3 lub **PRINT 2*3** <Enter> (musi być z przodu polecenie wydruku wyniku)

Uruchomienie programu QBasic:
QBASIC [Enter] lub QBASIC Nazwa_pliku {Enter}
Naciskamy Esc i można wpisywać program.
Zapis programu przez Alt F S lub Alt F A (sAve As).
Wejście do menu górnego przez Alt lub Alt i litera menu gornego.
Opuszczenie QBasic przez opcje eXit z menu File.
Program najlepiej wpisywać małymi literami.
Jesli jest to slowo kluczowe to po nacisnieciu Enter, slowo zostanie zamienione na duże litery.
Pozostałe słowa będą wpisane małymi literami.

Skróty klawiszowe w QBasic
Shift F1 - help
F6 - **tryb bezpośredni / tryb programowy** - przełączanie
Shift F5 - uruchomienie programu
F1 na słowie programu - pomoc do słowa
F4 - **ekran wynikowy** – by obejrzeć wyniki obliczeń

Typy **zmiennych** w QBasicu i ich deklarowanie (np. całkowity INTEGER jako DEFINT lub % na końcu zmiennej)
INTEGER – deklaracja przez **DEFINT** lub **%** na końcu nazwy - całkowite 2 bajtowe (2 * 8 = 16 bitów)
LONG INTEGER - deklaracja przez **DEFLNG** lub **&** na końcu - całkowite 4 bajtowe (4 * 8 = 32 bity)
SINGLE PRECISION - **DEFSNG** lub **!** na końcu - rzeczywiste pojedynczej dokładności. 4 bajtowe
DOUBLE PRECISION – **DEFDBL** lub **#** - rzeczywiste podwójnej dokładności. 8 bajtowe
STRING VARIABLE – **DEFSTR** lub **$** - zmienne typu znakowego

Przykłady:
DEFINT K, S ' deklarujemy zmienne typu integer na literę K i S [Enter],
DEFLNG L-N - wszystkie zmienne o nazwach zaczynających sie na litery z przedziału L do N (L, M, N lub l, m, n) będą traktowane jako zmienne typu długich liczb całkowitych.

Zmienna, której nazwa nie jest ujęta w żadnej deklaracji i nie jest zakończona żadnym ze znaków (%, &, !, # czy $), jest przez interpretator deklarowana jako zmienna typu SINGLE PRECISION - rzeczywista, pojedynczej dokładności, 4 bajtowa

Obliczenie wartości: ***3,14 razy (5 do kwadratu)*** w trybie bezpośrednim w QBasicu
Przechodzimy na tryb bezpośredni: **F6** i piszemy: **? 3.14 * 5^2 [Enter]**

Kasowanie ekranu w QBasic – **CLS** (jak w DOSie)

7. Komentarze w QBasicu: **‘** (apostrof() lub **REM** na początku komentarza
Np.
‘ Program P1.bas
REM progran P1.bas

Wprowadzenie danych: INPUT lub **LINE INPUT**
INPUT [;] ["prompt"{; | ,}] variablelist
należy rozumieć tak:
INPUT jest elementem, który musi wystąpić w instrukcji, po słowie INPUT możemy, ale nie musimy wpisać średnik (;), następnie możemy ale nie musimy wpisać dowolny tekst zamknięty cudzysłowami i zakończony średnikiem (;)

albo przecinkiem (,), na koncu musi sie znajdowac nazwa zmiennej lub kilka nazw rozdzielonych przecinkami.
Najprostsza instrukcja czytania danych z klawiatury ma postać: **INPUT** a1
Instrukcja: **INPUT liczba1, liczba2, liczba3**
zostanie wykonana po wpisaniu z klawiatury 3 liczb rozdzielonych przecinkami i wciśnięciu Enter.
W instrukcji INPUT powinno sie zawsze umieszczać informacje dla operatora, jakie wielkości ma wprowadzić.
Przykładem może być:
INPUT "Podaj 3 liczby z przedzialu 0 do 100"; n1, n2, n3

LINE INPUT służy do wprowadzenia jednej stałej znakowej, która może zawierać w sobie przecinki.
W instrukcji INPUT przecinek jest znakiem oddzielającym stale i nie może byc wczytany do zmiennej.
W instrukcji LINE INPUT dopiero [Enter] jest znakiem końca stałej i równocześnie sygnałem do jej przypisania do zmiennej umieszczonej w tej instrukcji.

Przykład: **INPUT "Podaj liczbe uczniow w klasie "; liczbauczniow%**
i wpisujemy (zamiast liczby) słowo trzydzieści.

Wydruk wyników na ekran: PRINT
Np. **PRINT** „Obliczenie azymutu”
PRINT x, y
PRINT USING <wyrażenie wyświetla łańcuchy lub liczby z użyciem określonego formatu
Np. **PRINT USING „####.##”; x**
- wydrukuje wartość zmiennej x, 4 miejsca na część całkowitą, 2 miejsca po przecinku

Tablice
DIM <lista> definiuje maksymalne wartości indeksów tablic i rezerwuje na nie odpowiednia ilość pamięci
Np. **DIM A(20)** lub **DIM cena(20,30) DIM cena(1 TO 1000) - liczby rzeczywiste zwykłej precyzji**
lub **DIM TXY#(2, 20)** - tablica 2 wymiarowa, podwójna precyzja bo # na końcu nazwy
DIM Nazwiska$(50 – rezerwacja tablicy na 50 nazwisk
DIM polozenie%(1 TO 10, 1 TO 20, 1 TO 30) – tablica na liczby całkowite, 3 wymiarowa

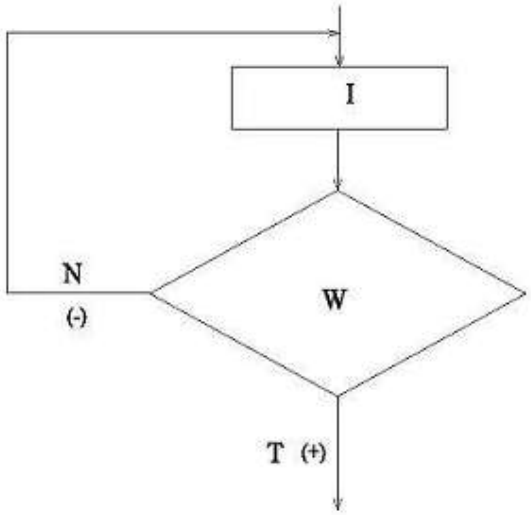
Pętla iteracji "dla" do powtarzania bloku instrukcji podaną ilość razy
FOR ... NEXT powtarza blok instrukcji podana ilość razy.
Składnia:
FOR <licznik> = <start> **TO** <koniec> [step]
lista_instrukcji
NEXT <licznik
Przyklad:
```
FOR k = 1 TO 20 STEP 1          ‘ step 1 oznacza krok 1, może byc pominięty
  PRINT k
NEXT k
```

Pętla iteracji "**Powtarzaj Instrukcja aż Warunek**"
Powtarzaj I aż W
Przynajmniej raz się wykona – warunek sprawdzany na końcu
Do Instrukcja **LOOP Until** Warunek
Lub
DO UNTIL Warunek
Instrukcja
LOOP

Np.
```
DO
  k = k + 1
  PRINT k
LOOP UNTIL k = 20
```

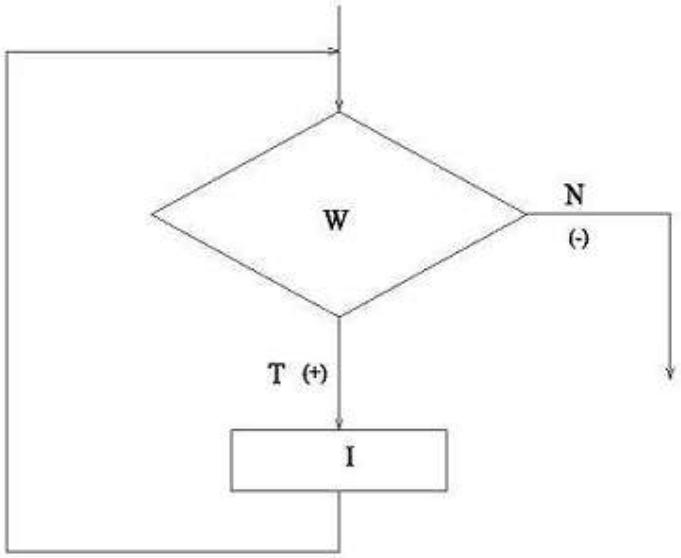
Pętla iteracji "Dopóki W wykonuj I"
Warunek sprawdzany na początku – może się ani raz nie wykonać
DO I LOOP WHILE W
Lub

DO WHILE W I LOOP
Lub
WHILE W I WEND

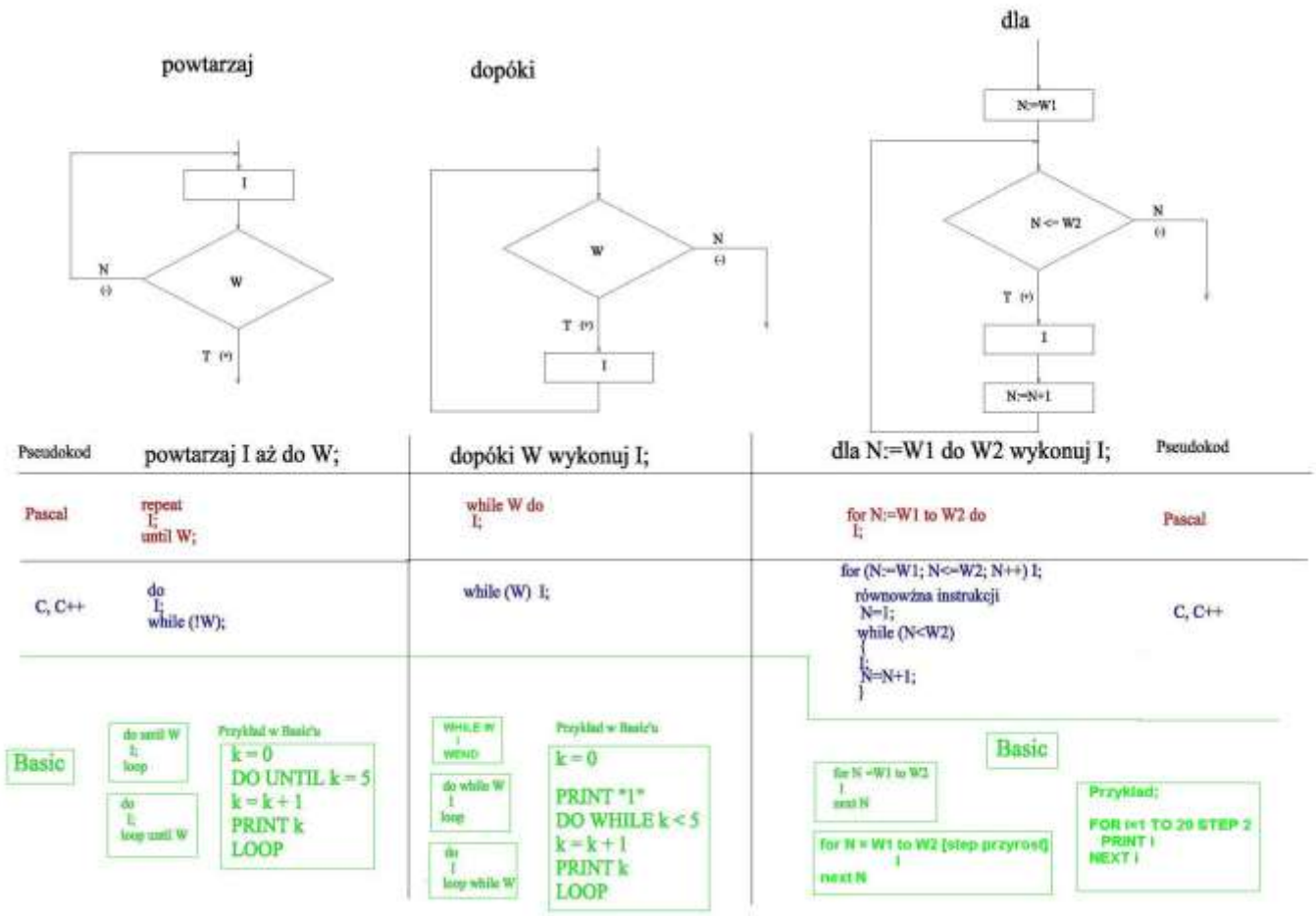


Pseudokod powtarzaj I aż do W;

dopóki



dopóki W wykonuj I;

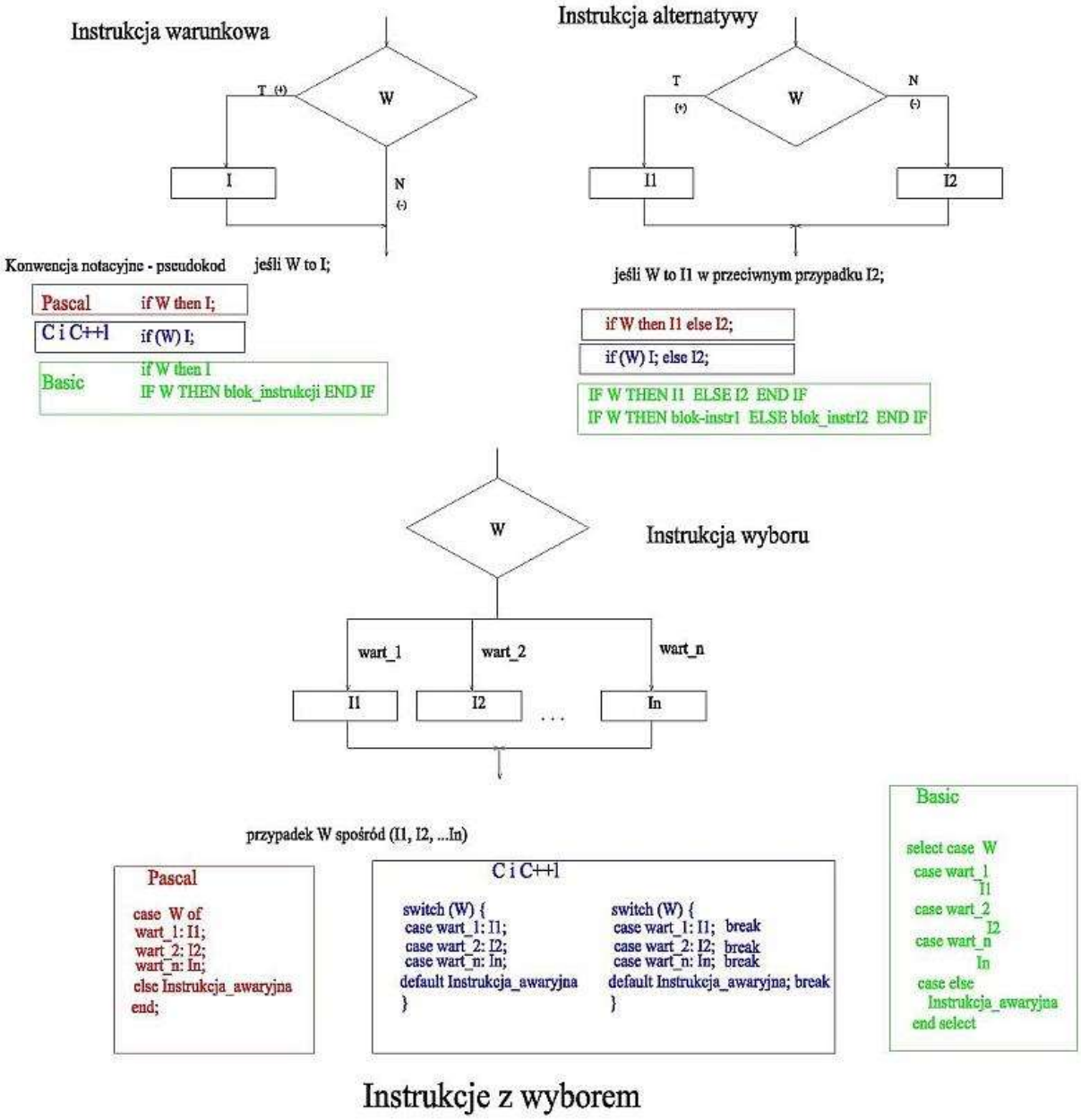


Instrukcje decyzyjne (z wyborem)

- ***Warunkowa:*** **IF W then I** lub **IF W THEN blok_instrukcji END IF**
  ‘ jeśli Warunek to Instrukcja

- ***Alternatywy*: IF W THEN I1 ELSE I2 END IF** ‘Jeśli W to I1 w przeciwnym razie I2

- ***Wyboru (CASE – przypadek z kilku):***
  SELECT CASE W **‘** Przypadek W
  CASE W1 I1 **‘** przypadek wartosc1 – instrukcja I1
  CASE W2 I2 **‘** -‘’- wartosc2 – instrukcja I2 itd.
  DEFAULT Instrukcja_awaryjna **‘** W przeciwnym razie instrukcja awaryjna
  END SELECT **‘ Koniec instrukcji CASE**



Instrukcje z wyborem

Zdanie decyzyjne - "Jeśli Warunek to Instrukcja" (if …)

Najprostsza forma instrukcji warunkowej jest:
IF warunek THEN instrukcja
w której warunek jest wyrażeniem logicznym
Przykład

```
INPUT "Podaj liczby a,b "; a, b
IF a<b THEN PRINT a;"<";b
IF a>b THEN PRINT a;">";b
```

IF warunek THEN blok END IF
Ten wariant umożliwia wykonanie całego bloku instrukcji w przypadku gdy warunek jest prawdziwy.

Przykład
```
IF a > b THEN
  wynik = a - b
  PRINT "a-b="; wynik
END IF
```

Funkcje operujące na znakach - wymień niektóre

CHR$ i ASC
Są to funkcje pozwalające na zamianę kodów ASII - CHR$ na odpowiadające im znaki
oraz na znajdowanie kodów ASCII podanych znaków - ASC.
CHR$(Numer_kodu)
Zwraca literę (łańcuch jednoznakowy) odpowiadającą danemu numerowi kodu ASCII.
Argumentem funkcji może być dowolny kod ASCII, liczba od 0 do 255. Wynikiem jest znak odpowiadający kodowi (liczbie).
Np. **print chr$(77)** da w wyniku **M**

ASC(wyrazenie_lancuchowe)
Zwraca nr kodu ASCII pierwszej litery w wyrażeniu łańcuchowym

Przykład 1:
```
start:    ' lub [start]
iNPUT "znak"; zn$  ‘ Wprowadzamy jakiś znak z klawiatury
PRINT ASC(zn4)              ‘ wydruk kodu znaku (liczby)
GOTO start
```

Przyklad 2
```
' Wydruk wszystkich znaków ASCII
CLS
PRINT "Kody ASCII i znaki "
FOR i = 0 TO 255
 PRINT "kod: ", i, "  znak ", CHR$(i)   ‘ Wyświetla znak (np. literę) dla kolejnych cyfr od 0 do 255
NEXT i
```

Inne funkcje operujące na znakach:
Np.

LEFT$(wyrazenie_lancuchowe, n) - Zwraca n pierwszych znaków wyrażenia łańcuchowego

RIFHT$(wyrazenie_lancuchowe, n) - Zwraca n ostatnich znaków wyrażenia łańcuchowego
np. **a$="dzien dobry" : print right$(as$, 5)** ‘ Na ekranie będzie dobry

MID$(wyrazenie_lancuchowe, start, [dlugosc]) - Funkcja zwraca łańcuch wycięty z wnętrza łańcucha
np. **print mid$("Ala ma kota", 1, 3)** - Wynik: Ala

LEN(wyrazenie_lancuchowe) ‘Zwraca ilość znaków w wyrazeniu łańcuchowym,

Czym różni się podprogram od funkcji?
Podprogram **SUB** (inaczej procedura) - może wykonać wiele operacji i nie zwraca konkretnej jednej wartości, funkcja FUNCTION zwraca jedną konkretną wartość, choć może też wykonać wiele operacji

Składnia podprogramu,
```
SUB  nazwa_ogoln (lista_parametrow_formalnych)
...
END SUB
```

Np.
```
SUB drukuj (x)
PRINT "Jestem w podprogramie DRUKUJ ";
```

```
PRINT "x="; x, "a="; a, "b="; b
END SUB
```

Deklaracja procedury na początku programu
DECLARE SUB nazwa_ogoln (lista_parametrow_formalnych)
Np. **DECLARE SUB drukuj (x!)** ' ! oznacza single precision – real, x! – parametr formalny

Wywołanie podprogramu
CALL nazwa_podprogramu (parametry_aktualne)
Np. **CALL drukuj(b) ‘ Wywołanie podprogramu z parametrem aktualnym b**

Funkcja,
Funkcja jest oddzielnym modułem programu, w którym mogą być wykonywane różne operacje, a ich wynik przekazywany do programu głównego poprzez nazwę funkcji.
Np.

```
FUNCTION sumprz (a, b, c)
  p1 = SQR(a ^ 2 + b ^ 2)
  p2 = SQR(b ^ 2 + c ^ 2)
  p3 = SQR(a ^ 2 + c ^ 2)
  sumprz = 2 * (p1 + p2 + p3)
END FUNCTION
```

Deklaracja funkcji: DECLARE FUNCTION nazwa(parametry)

Wywołanie funkcji: Nazwa (parametry_aktualne)
Chcąc w programie skorzystać z wartości obliczonej przez zdefiniowaną funkcję, posługujemy sie nazwą funkcji
Nazwa (parametry)
Np.

```
INPUT "Podaj 3 liczby "; a, b, c
PRINT "Suma dlug. przek prostop a,b,c= "; sumprz(a, b, c)
```

Inny przykład:

```
PRINT sumprz(1,2,5)
```

……..
Przedostatnia instrukcja by przekazać wynik działania funkcji do jej nazwy
musi być instrukcja **przypisania wyniku działania funkcji do jej nazwy.**

```
FUNCTION znaki$ (a$)
nl = LEN(a$)
FOR i = 1 TO nl
b$ = MID$(a$, i)
nb = ASC(b$)
IF nb >= 97 AND nb <= 122 THEN
  MID$(a$, i) = CHR$(nb - 32)
END IF
100 NEXT i
znaki$ = a$        ‘ Przedostatnia instrukcja
END FUNCTION  ‘ Ostatnia instrukcja
```

Druga metoda definiowania funkcji, którą można umieścić w dowolnym miejscu programu głównego
DEF FNnazwa

Składnia 1:
DEF FNnazwa[(zmiennne) = wyrazenie

składnia 2:
DEF FNnazwa[(zmienne)
....

FNnazwa = wyrazenie
[EXIT DEF]
...

END DEF

Operacje dyskowe

1) zapis wyników do pliku:

otwarcie pliku: OPEN plik FOR OUTPUT AS #kanał
np. **OPEN „wynik1.txt" FOR OUTPUT AS #1**
zapis danych:, **PRINT #1, dane**
zamknięcie pliku: **CLOSE #1**

2) Czytanie danych z pliku :

Otwarcie: OPEN plik FOR INPUT AS #kanał
Np. OPEN "dane1.txt" FOR INPUT AS #2
czytanie INPUT #kanał, dane
INPUT #1, x, y
zamknięcie.: CLOSE #kanał, np. **CLOSE #2**

Uruchomienie QBasica

QBASIC.EXE lub QBASIC nazwa_programu
Program musi mieć rozszerzenie BAS
Otwarcie – wczytanie pliku: File, Open
zapisać - File, Save
Uruchomienie: Run lub Shift F5

Rozszerzenie pliku w QBasicu:
BAS