**Chapter 4 Data structure**

**4.2 Array**

DATA:

* n: the number of initial values, integer, 1≤n<100, read from KBD
* n real values, read from KBD
* inserted values: real type, read from KBD
* insertion position k: integer type, 1≤k≤n

Result:

* an array of n+1 real values

**Algorithm** Insertion

**Variables:**

n: Integer

DataList: array [1..100] of real

k: Integer

index: Integer

**Instructions:**

Write (“Enter the number (n) of values in your datalist (1≤n<100): ” !)

Read (KBD! n)

For index From 1 to n Step 1

Write (“Enter your ”, index, “-th real value: ” !)

Read (KBD! DataList[index])

EndFor

Write (“Enter the position (k) of insertion (1≤k≤n): ” !)

Read (KBD! k)

For index From n to k Step -1

DataList[index+1] ← DataList[index]

EndFor

Write (“Enter the inserted value: ” !)

Read (KBD! DataList[k])

For index From 1 to n+1 Step 1

Write (DataList[index], “ ” !)

EndFor

**End** Insertion

**4.3 Limitations of an array**

It is not advisable to use an array. Because the number of values is arbitrary not a constant.

**Algorithm** PN\_Count

**Variables:**

P, N: Integer

nb: Real

flag: character

**Instructions:**

P ← 0

N ← 0

flag ← ‘N’

While flag=’N’ Do

Write (“Enter your new value: ” !)

Read (KBD! nb)

If nb≥0 Then

P ← P +1

Else

N ← N +1

EndIf

Write (“End of values ? (answer Y or N): ” !)

Read (KBD! flag)

EndWhile

**END** PN\_Count

It’s not possible to modify this algorithm in order to count the number o f values exceeding the mean value of all entered values.

The number of entered values must be known as a constant. Then we can use an array to memorize the list of data.

**4.6 Eliminating zeros**

**Algorithm** SuppZeros

**Variables:**

valarray: array [1..400] of real

dim, ind: Integer

k: Integer

**Instructions:**

Write(“Enter an integer please” !)

Read(KBD! dim)

For ind From 1 to dim

Write(“Enter a real number please” !)

Read(KBD! Valarray[ind])

EndFor

For ind From 1 To dim Step 1

If (valarray[ind]=0) Then

For k From ind To dim-1

valarray[k] ← valarray[k+1]

EndFor

dim ← dim -1

ind ← ind -1 // 避免连续出现两个0 的情况

EndIf

EndFor

For ind From 1 to dim

Write(valarray[ind], “ ” !)

EndFor

**End** SuppZeros

**4.7 Eratosthenes and prime numbers**

DATA:

* N: integer type, 2≤N≤100000, read from KBD

Result:

* A list of prime numbers lowers than N: array of integer, ≤N, to be printed on screen.

Set a list of required variables:

N: integer

E: array[1..N-1] of integer, for storing the number from 2 to N

P: array[1..N-1] of integer, for storing the prime numbers <= N

ind, k: Integer

dim : Integer

nP: Integer

**Algorithm** Eratosthenes

**Variables:**

N, ind, k, dim, nP: Integer

E, P: array [1..400] of Integer

**Instructions:**

Write(“Enter an integer (2<= N <= 100000) please” !)

Read(KBD! N)

For ind From 1 to N-1

E[ind] ← ind + 1

EndFor

dim ← N-1

nP ← 0

While (dim>0) Do

nP ← nP +1

P[nP] ← E[1]

For ind From 1 to dim-1 // 去掉首项，首项肯定是最小的

E[ind] ← E[ind+1]

EndFor

dim ← dim -1

For ind From 1 to dim

If (E[ind]%P[nP]=0) Then

For k From ind To dim-1

E[k] = E[k+1]

EndFor

dim ← dim -1

ind ← ind -1 // 避免连续出现两个0 的情况

EndIf

EndFor

EndWhile

For k From 1 to nP

Write(P[k], “ ” !)

EndFor

**End** Eratosthenes