**5.5 Complex numbers**

1.

Types:

CplxCartesian: article (

Re: Real

Im: Real

)

CplxPolar: article (

Mod: Real

Arg: Real

)

2.

Types:

Cplx: article (

sys: Character // ‘C’ for Cartesian, ‘P’ for polar

param1: Real

param2: Real

)

3.

以 *RealPart* 为例：

**Sub\_Algorithm** *RealPart*

**Input**:

nb: Cplx

**Output**:

nb\_Re: Real

**Variables**:

**Instructions**:

If(Cplx.sys = ‘C’) Then

nb\_Re ← Cplx.param1

Else

nb\_Re ← Cplx.param1 \* cos(Cplx.param2)

EndIf

**End** *RealPart*

4.

**Sub\_Algorithm** *NearestNeighbors*

**Input**:

V: arry[1..] of Cplx

nb: Cplx

**Output**:

Index: Integer

**Variables**:

temp: Cplx

distanceMin, distance: Real

k: Integer

**Instructions**:

*Subtract*(nb, V[1] ! temp)

*Modulus*(temp ! distanceMin)

Index ← 1

k ← 2

While(V[k].sys = ‘C’ OR V[k].sys = ‘P’) Do

*Subtract*(nb, V[k] ! temp)

*Modulus*(temp ! distance)

If(distance < distanceMin) Then

Index ← k

distanceMin ← distance

EndIf

k ← k + 1

EndWhile

**End** *NearestNeighbors*

**6.1 Books – Sort**

6.1.1

**Sub\_Algorithm** *OrderTEXTs*

**Input**:

string1, string2: array[1..] of Character

**Output**:

flag: Integer // 0: same; 1: string1 comes first; 2: string2 comes first

**Variables**:

k: Integer

**Instructions**:

k ← 1

flag ← 0

**While**(string1[k] ≠’\0’ AND string2[k] ≠’\0’ AND flag = 0) Do

If(string1[k]≥’A’ AND string1[k]≤’Z’ AND string2[k] = string1[k]+32)

flag ← 1

ElseIf(string2[k]≥’A’ AND string2[k]≤’Z’ AND string1[k] = string2[k]+32) Then

flag ← 2

Else

If(string1[k]≥’A’ AND string1[k]≤’Z’) Then // 大写字母转小写字母

string1[k] ← string1[k] + 32

EndIf

If(string2[k]≥’A’ AND string2[k]≤’Z’) Then // 大写字母转小写字母

string2[k] ← string2[k] + 32

EndIf

If(string1[k] < string2[k]) Then

flag ← 1

ElseIf(string1[k] > string2[k]) Then

flag ← 2

EndIf

EndIf

k ← k + 1

**EndWhile**

If(flag = 0) Then

If(string1[k] = ‘\0’ AND string2[k] ≠‘\0’) Then

flag ← 1

ElseIf(string1[k] ≠ ‘\0’ AND string2[k] = ‘\0’) Then

flag ← 2

EndIf

**End** *OrderTEXTs*

6.1.2

Types:

TLivre: article (

Title: array[1..100] of Character

Authors: array[1..5, 1..30] of Character

Edition: Integer

EditionYear: Integer

ISBNcode: array[1..17] of Character

rentable: Boolean

isBorrowed: Boolean

returnDate: array[1..8] of Integer // yyyymmdd

)

**Sub\_Alogrithm** SortBooks

**Input**:

Books: array[1..] of TLivre

**Output**:

BooksSt: array[1..] of TLivre

**Variables**:

*step*, i, N, flag: Integer

*permut*: Boolean

*inter*: TLivre

**Instructions**:

N ← 1

While (Books[N].Title[1] ≠ ‘\0’) Do

N ← N + 1

EndWhile

N ← N 1 // 书的数量

*permut* ← True

*step* ← 1

While(*step* <= N-1 AND *permut* = True) Do

*permut* ← False

For *i* From 1 To N-*step*

*OrderTEXTs*(Books[*i*].Author[1, 1..], Books[*i+*1].Author[1, 1...] ! flag)

If(flag = 0) Then

*OrderTEXTs*(Books[*i*].Title, Books[*i+*1].Title ! flag)

EndIf

If(flag = 2) Then

*inter* ← Books[*i*]

Books[*i*] ← Books[*i*+1]

Books[*i*+1] ← *inter*

*permut* ← True

EndIf

EndFor

*step* ← *step* + 1

EndWhile

For *i*From 1 To N

BooksSt[*i*] ← Books[*i*]

EndFor

**End** SortBooks

**6.2 Strange Sort**

**Algorithm** StrangeSort

**Variables**:

k: Integer

list: array[1..1000] of Integer

listSt: array[1..1000] of Integer

max\_posi, min\_posi: Integer

max\_val, min\_val : Integer

**Instructions**:

*getArray*( ! list, N) // 给数组list里读入数据，比如通过键盘读入等，

// 返回数组及其长度N

k ← 1

While(k≤N) Do

If(k%2=1) Then

*MinFunction*(list ! min\_val, min\_posi) // 输入一个数组，

// 返回最大值及所在位置

ListSt[k] ← min\_val

*EliminateFunction*(list, min\_posi ! list) // 输入一个数组

// 删除特定位置的元素

Else

*MaxFunction*(list ! max\_val, max\_posi) // 输入一个数组，

// 返回最大值及所在位置

ListSt[k] ← max\_val

*EliminateFunction*(list, max\_posi ! list)

EndIf

k ← k + 1

EndWhile

**End** StrangeSort

**7.2 Fibonacci**

**Sub\_Algorithm** *FibR*

**Input**:

n: Integer

**Output**:

Value: Integer

**Variables**: (non 无)

**Instructions**:

If(n=1) Then

Value ← 1

ElseIf(n=2) Then

value ← 2

Else

value ← *FibR*(n1 !) + *FibR*(n)

EndIf

**End** *FibR*

**Algorithm** Main

Variables:

n: Integer

fibn: Integer

Instructions:

Write(“Enter a positive integer n: ” !)

Read(KBD ! n)

*FibR*(n ! fibn)

Write(“The ” ,n, “-th Fibonacci element is: ”, fibn !)

**End** Main

**7.5 Search an element in a sorted list**

**Sub\_Alogrithm** *SearchSorted*

**Input**:

list: array[1..] of Real

target: Real

**Output**:

flag: Boolean

position: Integer

**Variables**:

list\_temp: array[1..] of Real

N: Integer

**Instructions**:

flag ← False

position ← 0

*Length*(list ! N) // return the length N of list

If(N = 1) Then

If (list[1] = target) Then

flag ← True

position ← 1

EndIf

Else

*Copy*(list, 1, N-1! list\_temp) // 把list的1 至 N-1个元素复制给list\_temp

*SearchSorted*(list\_temp, target! flag, position)

If(flag = False) Then

If (list[N] = target) Then

flag ← True

position ← N

EndIf

EndIf

EndIf

**End** *SearchSorted*

**Algorithm** Main

Variables:

list: array[1..1000] of Real

target: Real

N, position: Integer

flag: Boolean

Instructions:

Write(“Enter the sorted list: ” !)

*getArray*( ! list, N) // 给数组list里读入数据，比如通过键盘读入等，

Write(“Enter the target: ” !)

Read(KBD! target)

*SearchSorted*(list, target ! flag, position)

If(flag = True) Then

Write(“The target is found at position ”, position !)

Else

Write(“The target is not found.” !)

EndIf

**End** Main