Fortran Interface Components :

1- Title bar
2- Menu bar : file – view – build

variables naming Conditions:

1- Variable name must begin with a letter (not number or symbol) .
2- Variable name can’t include a space between the sentence and instead that the program use the underscore (F95 , F_95).
3- Variable name should not overtake 30 characters .
4- Fortran is sensitive to characters case , not distinguishing between the case of large or small letters (A = 6, a = 6).

Data types :

1- Integer : (-1 , 0 , 1)
2- Real : (1.4 , -1.4 , 1.4e2 . 1.4e-2)
3- Complex : (real numbers) and (imaginary part) == Z = X + Yi , for example : Complex (2.0 , -1.0) === 2.0 – 1.0i
4- Logical : (. true . and . false .)
5- Characters : (‘signal quotes’ , “double quotes”)

define variables in fortran :

```
Integer :: hours
Real :: Temp
Integer :: hour , minute , second
Real :: temp , dew_point , wind_speed , total , averages
Character ( len = 20) :: name
Character ( 20 ) :: name , line
Character :: first_initial*10
 Real , parameter :: pi = 3.14
Complex :: cx
Logical :: done
Integer :: total
Total = 7.6
Average  = average2
Done  = . true .
Line = “this is a line “
Cx = (1.0 , 2.0 ) ! 1.0 + 2.0i
Cx = cmplx ( 1.0/2.0 , -3.0 ) ! cx = 1.0 + 2.0i
```

Cx = cmplx (x , y) ! cx = x + yi

Note :

- (c , j , k , l , m , n) are integers and the rest are real
- (!) this symbol is use to add a note on the program steps
- (&) this symbol is use to continue the line , for example :
  Cos (alpha) = b*b – c*c - & 2*b*c*cos(gama)

Arithmetic operation in Fortran :

Example	Result	Symbol
2**3	Raise to power	**
2*3	Multiply	*
2/3	Divide	/
2+3	addition	+
2-3	subtraction	-

Priorities of Math Operations in Fortran:

1- Arches
2- powers (from right to left)
3- Multiplication and division
4- Addition and subtraction

Homework :: find that [(4 + 8**2) / 2 , 2**2**3 , 3*(1+2) , 3*2+1]

Input and Output :

1- Input : input list , read*
2- Output : print*, result_list or write (*,*)

Example 1 :

```
Program sum
! Example of program structure
Implicit none
```

Example 2 :

```
Program io
Real :: x , y , z
Print *, 'enter the values x , y and z'
Read *, x , y , z
Print *, ' the values you type are for z , y , x are : '
```

```
Real :: answer , x , y
Print *, ‘enter two numbers’
Read *, x
Read *, y
Answer = x+y
Print *, ‘ the total is ‘ , answer
End program sum
```

Example 3 :

```
Program bug
Real :: a , b , c
Read *, b , c
a = b+c
print *, a
end program bug
```

Homework : write a Fortran program to compute the equations :

1- $\frac{x+y}{x+2}$
2- xyz
3- xy^2
4- $\sqrt{x^3 + y^3}$
5- $\sin x \cos y$

Mathematical functions :

1- Exponential Functions

Example	Function in fortran form	Operation
>> exp(0) 1	exp(x)	Exponential function
>> log(1)	log(x)	Natural logarithmic function

0		
>> log10(2) 0.3010	log10(x)	logarithmic function base 10
>> log (2) 1	log (x)	logarithmic function
>> sqrt(4) 2	Sqrt(x)	Square rote function
Abs(5) 5	Abs(x)	Absolute function

2- Trigonometric Functions :

Fortran forms	Type of argument	Type of result
sin(angle)	real	real
cos(angle)	real	real
tan(angle)	real	real
atan(angle)	real	real
sinh(angle)	real	real
cosh(angle)	real	real
tanh(angle)	real	real

Note : All of the trigonometric functions above are measured by radian deg. In order to converting from degrees to radian, multiply the function by 180 (pi)

Asin(x)	Real $-1 \leq x \leq 1$	Real $-\frac{\pi}{2} \leq result \leq -\pi/2$
Acos(x)	Real $-1 \leq x \leq 1$	Real $0 \leq result \leq \pi$

Complex numbers :

Take a single formula (real number) and (imaginary number) : z = x + yi

Complx (x , y) : this function is use to define the complex number in fortran

Example :

Program dd

```
complex::cn
cn=cmplx(5.0,8.9)
print*,cn
end program dd
```

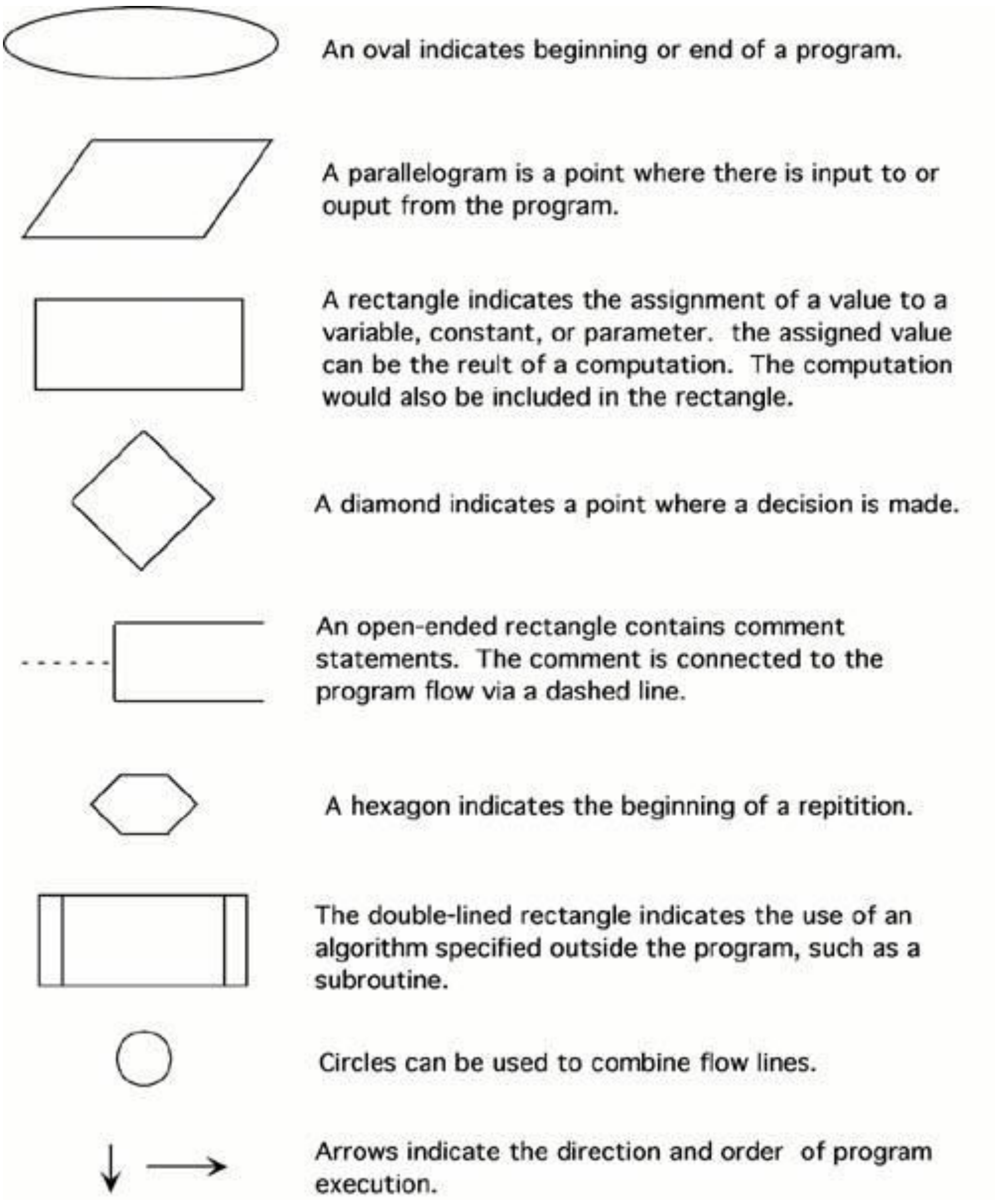
Routing & Remainder functions:

1- **floor** : Round toward negative infinity (- ∞) : floor (-3.4) = -4 , floor (3.4) = 3
2- **int** : Converts any number to an integer : int(0.3)=0 , int(-0.3)=0 , int(3.9)=3
3- **nint** : Round to nearest integer : nint(5.9)=6 , nint(-5.9)=-6
4- **real** : Convert number to real : real(-1.5)=-1.5000 , real(8)=8.000
5- **mod** : Modulus after division : mod(a,b) = a-int(a/b)*b :: mod(4,2)=0 , mod(9,4)=1
6- **modulo** : Remainder after division : modulo(a,b)== a-floor(a/b)*b :: modulo(8,10)=8 , modulo(-1,20)=-1

homework : Mod(6,10) , int(4.3) , floor(5.4) , nint(6.9) , modulo(8,10) , real(2.4) , nint(-3.4) Mod(5,8) , int(6.8) , floor(5.5) , nint(9.6) , modulo(8,10) , real(9/9) , nint(-3.5)

Flowcharts :

Flowcharts are written with program flow from the top of a page to the bottom. Each command is placed in a box of the appropriate shape, and arrows are used to direct program flow. The following shapes are often used in flowcharts:



An oval indicates beginning or end of a program.

A parallelogram is a point where there is input to or ouput from the program.

A rectangle indicates the assignment of a value to a variable, constant, or parameter. the assigned value can be the reult of a computation. The computation would also be included in the rectangle.

A diamond indicates a point where a decision is made.

An open-ended rectangle contains comment statements. The comment is connected to the program flow via a dashed line.

A hexagon indicates the beginning of a repitition.

The double-lined rectangle indicates the use of an algorithm specified outside the program, such as a subroutine.

Circles can be used to combine flow lines.

Arrows indicate the direction and order of program execution.

Pseudocode : is a method of describing computer algorithms using a combination of natural language and programming language. It is essentially an intermittent step towards the development of the actual code. It allows the programmer to formulate their thoughts on the organization and sequence of a computer algorithm without the need for actually following the exact coding syntax. Although pseudo code is frequently used there are no set of rules for its exact implementation. In general, here are some rules that are frequently followed when writing pseudo code:

1. The usual Fortran symbols are used for arithmetic operations (+, -, *, / , **).

2. Symbolic names are used to indicate the quantities being processed.

3- Certain Fortran keywords can be used, such as PRINT, WRITE, READ, etc.

4- Indentation should be used to indicate branches and loops of instruction.

Here is an example problem, including a flowchart, pseudocode, and the final Fortran 90 program. This problem and solution are from Nyhoff, pg 206:

For a given value, Limit, what is the smallest positive integer Number for which the sum

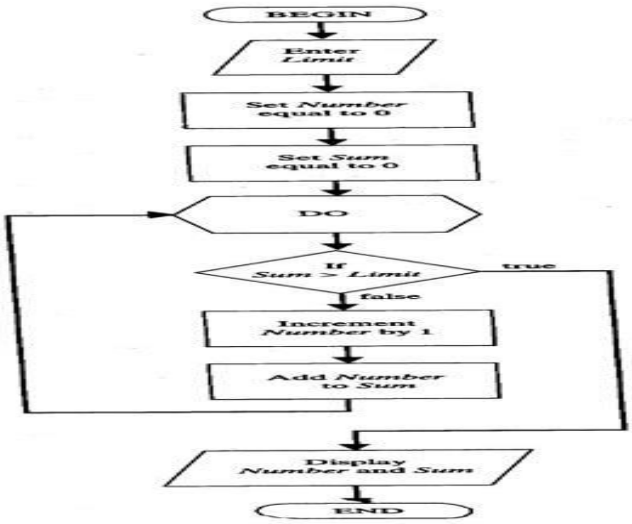
Sum = 1 + 2 + ... + Number

is greater than Limit. What is the value for this Sum?

Pseudocode:

```
Input:    An integer Limit
   Ouput:   Two integers: Number and Sum
    1. Enter Limit
    2. Set Number = 0.
    3. Set Sum = 0.
    4. Repeat the following:
       a. If Sum > Limit, terminate the repitition, otherwise.
       b. Increment Number by one.
       c. Add Number to Sum and set equal to Sum.
    5. Print Number and Sum.
```

Flowchart:



Fortran 90 code:

```
PROGRAM  Summation
!  Program to find the smallest positive integer Number
   !  For which Sum = 1 + 2 + ... + Number
   !  is greater than a user input value Limit.
IMPLICIT NONE
!  Declare variable names and types
INTEGER :: Number, Sum, Limit
!  Initialize Sum and Number
Number = 0
   Sum = 0
!  Ask the user to input Limit
PRINT *, "Enter the value for which the sum is to exceed:"
   READ *, Limit
!  Create loop that repeats until the smallest value for Number is found.
DO
      IF (Sum > Limit) EXIT    !  Terminate repetition once Number is
found
      !  otherwise increment number by one
      Number = Number + 1
      Sum = Sum + 1
   END DO
```

```
!  Print the results
PRINT *, "1 + ... + ", Number, "=", Sum, ">", Limit
END PROGRAM Summation
```

Homework : write F95 program to solve and draw flowchart of example (1-2-3)