**1/08/14**

# Chapter 1:

## Binary numbers

10010100

^

2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

128 + 0 + 0 + 16 + 0 + 4 + 0 + 0 = 148

231(10) -> 11100111(2)

Source code 🡪 compiler (g95) 🡪 executable file

! this is a comment

PROGRAM ex1

WRITE(\*,\*)”Hello”

WRITE(\*,\*)”world”

END PROGRAM

**1/15/14**

# Chapter2: Basic Elements

Variable names: 0-9, \_, a-z, A-Z

Ex1

## Other variable types

INTEGER

REAL

CHARACTER (len=20)

LOGICAL .TRUE. .FALSE.

COMPLEX

Ex2

Ex3

+ addition

- subtraction

\* multiplication

\*\* raise to the power

/ divide

Integer math

WRITE(\*,\*) 2 /3 ! output is zero

WRITE(\*,\*) 2 / 3.0 ! output is .666

## Order of operations

\*\* from right to left

\* / from left to right

- + from left to right

WRITE(\*,\*) 1.0 + 3.6 / 1.2 – 8.0 ! output is -4.0

Intrinsic functions p48

SQRT(X)

SIN(X)

COS(X)

Ex4

Homework: 2.(1,3,4) DUE NEXT Wednesday

**1/22/14**

# Chapter 3: Branching Statements

Program design

Top down design

Flow charts

## Comparison Operators

< less than

> greater than

<= less than or equal to

>= greater than or equal to

/= not equal to

== equal

3 > 4 .FALSE.

3 < 5 .TRUE.

21 <= 21 .TRUE.

5 /= 12 .TRUE.

7 == 8 .FALSE.

## Combinational logic operators

.AND. and operator

.OR. or operator

.EQV. if both true or both false to be true

.NEQV. both cannot be same to be true

.NOT. negates the value

X = 2.1

Y = 5.8

X >= 0.0 .AND. Y < 10 .TRUE.

Y == 13.0 .OR. X /= 0 .TRUE.

## The IF statement

IF ( logical expression ) THEN

… code

… code

END IF

PROGRAM Ex1

IMPLICIT NONE

REAL :: a, b, c, r1, r2

WRITE(\*,\*) “Welcome to the quadratic roots solver”

WRITE(\*,\*) “ Enter values for a, b, and c”

READ(\*,\*) a, b, c

IF ( b\*\*2 – 4\*a\*c < 0) THEN

WRITE(\*,\*) “Your roots are imaginary.”

STOP

END IF

r1 = (b + SQRT(b\*\*2 – 4\*a\*c)) / (2\*a)

r2 = (b - SQRT(b\*\*2 – 4\*a\*c)) / (2\*a)

WRITE(\*,\*) “Your roots are “, r1, “ and “, r2

STOP

END PROGRAM

IF THEN ELSE

IF ( logical expression ) THEN

… code

… code

ELSE

… code

… code

END IF

WRITE(\*,\*) “Enter your grade”

READ(\*,\*) grade

IF (grade >= 70) THEN

WRITE(\*,\*) “you might pass”

ELSE

WRITE(\*,\*) “put some effort into it”

END IF

## Single line IF statement

IF ( logical expression ) command … code

LOGICAL :: error

error = .FALSE.

IF ( error .EQV. .TRUE.) STOP

IF ( error ) STOP

More examples

PROGRAM Ex2

IMPLICIT NONE

INTEGER :: grade

WRITE(\*,\*) “welcome to grade analysis 0.00.1”

WRITE(\*,\*) “what is your grade?”

READ(\*,\*) grade

IF (grade >= 90) THEN

WRITE(\*,\*) “you got an A!”

END IF

IF (grade >= 80 .AND. grade < 90) THEN

WRITE(\*,\*) “you got an B.”

END IF

IF (grade >= 70 .AND. grade < 80) THEN

WRITE(\*,\*) “you got an C.”

END IF

IF (grade >= 60 .AND. grade < 70) THEN

WRITE(\*,\*) “you got an D.”

END IF

IF (grade < 60) THEN

WRITE(\*,\*) “you got an F.”

END IF

# Chapter 4: Loops

## The while loop

DO

.. code ..

IF (logical statement) EXIT

.. code ..

END DO

I = 1

DO

WRITE(\*,\*) I, “ \* “, I, “ = “, I \* I

IF (I >= 10) EXIT

I = I + 1

END DO

1 \* 1 = 1

2 \* 2 = 4

..

9 \*9 = 81

10 \* 10 = 100

WHILE

..code..

IF (logical statement) EXIT

..code..

END of WHILE

Factorial Example

N! = (n) \* (n-1) \* (n-2) \* …

And 0! = 1

PROGRAM ex1

IMPLICIT NONE

INTEGER :: n, fact

WRITE(\*,\*) “enter n to calculate the factorial”

READ(\*,\*) n

fact = 1

I = 1

DO

IF (I > n) EXIT

fact = fact \* 1

I = I + 1

END DO

WRITE(\*,\*) “The factorial is “, fact

END PROGRAM

## Counting do loops

DO counting variable = start, ending, iterator

..code..

..code..

END DO

INTEGER :: o

DO o = 1, 10

WRITE(\*,\*) I, “ \* “, I, “ = “, I \* I

END DO

DO j = 1, 20, 2

WRITE(\*,\*) j

END DO

DO k =10, 1, -1

WRITE(\*,\*) k

END DO

DO f = 10, 20, -1

WRITE(\*,\*) f

END DO

! does nothing

I = 1

DO

IF (I > 10) EXIT

WRITE(\*,\*) I

I = I – 1

END DO

! infinite loop

# Chapter 5: Basic IO

## Format of READ and WRITE

Our complier does not does control characters (pg 186)

I desctiptor

rIw

F desctiptor

rFw.d

INTERGER :: I = 12

REAL :: x = 2.7189

WRITE(\*,24) I, j

24 FORMAT( “I = “, I7, “ x = “, F6.3)

I = 12 x = 2.719

E descriptor

rEw.d

X descriptor

rX

x = 234.5824

WRITE(\*,48)

48 FORMAT(4X, E14.4)

0.2346E+03

ES desctiptor

rESw.d

x = 234.5824

WRITE(\*,48)

48 FORMAT(4X, E14.4)

2.3458E+02

A descriptor

rQw

for characters

CHARACTER(len=12) :: name = “bob”

WRITE(\*,88) name

88 FORMAT(2X,A8) name

Bob

## Read and Write to a file

OPEN command

OPEN(UNIT = 19, FILE = “filename”, STATUS = ‘OLD’, ACTION = ‘READ’, IOSTAT = var)

PROGRAM ex1

IMPLICIT NONE

INTEGER :: I, j, count, err1, err2

REAL :: num

OPEN(UNIT = 19, FILE = “data\_in.dat”, STATUS = ‘OLD’, ACTION = ‘READ’, IOSTAT = err1)

OPEN(UNIT = 44, FILE = “data\_out”, STATUS = ‘UNKNOWN’, ACTION = ‘WRITE’, IOSTAT = err2)

IF (err1/=0 .OR. err2/=0) THEN

WRITE(\*,\*) “error in file open”

STOP

END IF

Count = 0

DO

READ(19,\*,IOSTAT=err1) num

IF (err1/=0) EXIT

Count = count + 1

WRITE(44,\*) num \* 1000

END DO

CLOSE(19)

CLOSE(44)

END PROGRAM

**Homework CH5:1,3,5**

# Chapter 6: Arrays

Bubble sort

DO I = 1, count -1

DO j = 1, count -1

IF (ar1(j)>ar1(j+1)) THEN

Tmp = ar1(j)

Ar1(j) = ar1(j+1)

Ar1(j+1) = temp

END IF

END DO

END DO

WRITE(\*,\*) “your data, sorted”

DO I = 1, count

WRITE(\*,\*) ar(i)

END DO

CLOSE(33)

END PROGRAM

! whole array operations

INTEGER, DIMENSION(10) :: I1 ,I2, I3

INTEGER :: I, j

I1 = (/1,2,3,4,5,6,7,8,9,10/)

I2 = (/10,9,8,7,6,5,4,3,2,1/)

! add together

DO I = 1, 10

I3 = i1(i) + i2(i)

END DO

WRITE(\*,\*) i3

I3 = i1 + i2

WRITE(\*,\*) i3

I3 (6:10) = i1(1:5) + i2(3:7)

**Homework Ch6 : 2,4,5**