Ho Chi Minh City University of Natural Sciences

Faculty of Physics - Technical Physics

Department of Oceanography, Meteorology and Hydrology

APPLICATION PROGRAMMING

MSc. Nguyen Hoang Phong MSc.

Nguyen Tien Thanh

Objectives of the subject

- Apply programming languages in calculations
- Generate ideas to solve problems
- Determine the appropriate algorithm
- Have independent and creative thinking

Learning outcomes of the subject

- Understand some commands and structures in Fortran -

Apply Fortran programming language in data processing and calculating simple problems - Apply some

drawing tools and present results

- Generate ideas to solve problems

- Determine the appropriate algorithm

- Have independent and creative thinking

Subject content

I. Support software

- Surfer

- Grapher

- WRPLOT view

II. Basic Fortran

- Fortran programming

language - Numerical

recipes III. Toolbox in matlap

Time distribution

1) General introduction

2) Fortran programming language

3) Fortran programming language (cont.)

4) Practice fortran programming language

5) Practice Fortran programming language (cont.) 6)

Apply Fortran programming to solve problems (gk) 7)

Apply Fortran programming to solve problems (cont.)

8) Numerical recipes fortran

Time distribution (cont.)

9) Practice Sufer

10) Practice grapher, WRPLOT view (ed) 11)

Review matlap

12) Toolbox in matlap 13)

Toolbox application in practice (cont.) 14)

Review

15) Review (cont.)

Document

- Phan Van Tan (2007), Fortran 90 programming language, Hanoi National University Publishing House.

- Fortran 90 User's Guide

- Brian D Hahn (1994), FORTRAN 90 for Scientists and Engineers, Butterworth Heinemann

Regulations

- Students need to strictly comply with the rules and regulations of the Faculty and School.
- Students are not allowed to miss more than 3 sessions out of the total number of theory sessions. If they miss more than 3 sessions, they will be banned from taking the

exam. • Students come to class on time, but students who come in late get no poi

name.

- For any cheating during assignments or exams, students must be subject to all forms of discipline by the Faculty/ School and receive 0 points for this subject.

Point structure

- Final exam: 50% (LT (open): 30%, TH: 20%) •

Midterm exam: 20%

- Small exercises:

20% • Attendance: 10%

FORTRAN INTRODUCTION

Introduction

1) Programming language: is a notated system for describing calculations (through a computer) in a form that both humans and machines can read and understand.

- Easy to understand and use for programmers, so it can be used to solve many different problems.

- Describe processes completely and clearly , to run on other computer systems

together.

Every programming language is also a program, but it can be used to create other programs . The text written in a programming language to create a program is called source code.

Programming languages are a subset of computer languages, designed and standardized to transmit instructions to machines with processors (CPUs), in particular computers . Programming languages are used to program computers , create machine programs for the purpose of controlling computers or describing algorithms for others to read and understand.

Machine language is instructions in binary form, directly intervening in electrical circuits

death.

- Programs written in machine language can be executed immediately without any intermediate steps.

- However, programs written in machine language are error-prone ,

cumbersome, and difficult to read and understand because they are full of nu
0 and 1.

Assembly language was a giant step forward in moving programming languages away from confusing machine language. Appearing in the 1950s, it was designed to make computers more user-friendly.

- Assembly language introduces the concept of variables.
- Assembly language also contains a few “pseudo-operations”, which can represent the operation code as statements (also known as commands) instead of in binary form. Commands consist of two parts: the code part (written in English title) indicates the mathematical operation to be performed and the variable name part indicates the address containing the operand of the c That math.
- In order for a machine to execute a program written in assembly language, the program must be translated into machine language.

The tool that does that translation is called Assembler

High level language is a language created and developed to reflect the way programmers think and work.

- High level language is very close to human language (English) but as precise as mathematical language. Thanks to high-level languages, programming has become popular , many people can write programs, and thanks to that, software has developed rapidly, serving many areas of life.

- A program written in a high-level language is called a source program. In order for the computer to "understand" and execute the instructions in the source program, there must be a compiler program to translate the source program (written in a high-level language) into a target program.

A program written in a high-level language must use a translator to convert it into a machine language program to be able to execute.

Programming in a high-level language is easier to write because the instructions are encoded closer to natural language.

Programming in machine language is very difficult, usually only programming experts can do it.

2) Programming (computer programming, often referred to as programming) is the use of data structures and commands of a specific programming language to describe data and express algorithmic operations.

3) Components of programming languages

Alphabet

-A set of symbols used to write programs
-From AZ

Syntax

- A set of rules used to write programs
- How to write a valid program
- **Write (*,*) a**

Semantic

- Determine the meaning of character combinations in the program

4. Some concepts

4.1 Name: Every object in the program must be named according to the rules of the programming language and each specific translation
program - **Reserved names:** These are names specified by the programming language with a
specific meaning that programmers cannot use with other meanings. The reserved name, when spelled correctly, will appear
- **Name given by the programmer:** Determined by declaration before use and must not overlap with reserved names; The names in the program cannot be the same

4.2 Constants

and variables Constants: Are quantities that have constant values during

program execution **Variables:** Are named quantities whose values can be changed during the program

- **Brief history of the fortran programming language**

- A set of special rules for encoding knowledge for a computer to understand is called a programming language submit.

- John Backus proposed around the end of 1953 in April New York → 1957. - 1966, first standard version.

- Fortran 77 (1978) → Fortran 90 (1991) → Fortran 95, Fortran 2003, Fortran 4.0, Fortran 6.0,….

- Fortran can work on many different operating systems, such as UNIX, LINUX, WINDOWS, DOS,...

Structure of a basic fortran program

```
Program crystal_do_cao_song
integer*4 n, d
Parameter (n=2000,d=23)
integer*4 i,j,m,k
real*4 a(n),b(n),c(n),h(n) ,tam,e(d),f(d),x(d)
Open (unit=1,file='KQKT_cs_00_0902.txt',status='old')
-----------------------------------------
Close (1)
End
Subroutine tb(a,n,tbc)
-----------------------------------------
Endsubroutine
```

- Type program lyrics *(source code)*
- Run *(run)*
- Compile : translate *into* machine *code*

+ Check syntax *(Syntax)*
+Translate *into*
computer code + Object
+ Link *(Link)*
+ *Executable file*
- *Executed*

How to name variables and constant names

- Can contain 1 to 31

characters • Must begin with an English letter. The characters used to construct variable and constant names consist of 26 English letters

- Does not distinguish between lowercase and uppercase letters

(AÿZ and aÿz), 10 digits (0ÿ9), and *underscores* (_)

For example:

a) 2A b)

An An c)

delta_4x d)

theta+alpha

Declare the data type for the variable

• 5 standard data types  double layer

+ Class of numeric *types*

- Integer **(integer)**

- Real numbers **(real)**

- Complex numbers **(complex)**

+ Class of non-numeric types *(nonnumeric)*

- Character type **(character)**

- Logical type **(logical)**

Class of numeric types (Integer, Real, Complex)

- Integer types are data that accept values belonging to the set of integers

How to declare	Number of bytes occupied	Value range
INTEGER	4	**ÿ2147483 648 to 2147483 647**
INTEGER*1 or **INTEGER (1)** or **INTEGER (KIND=1)**	first	**ÿ128 to 127**
INTEGER*2 or **INTEGER (2)** or **INTEGER (KIND=2)**	2	**ÿ32 768 to 32 767**
INTEGER*4 or **INTEGER (4)** or **INTEGER (KIND=4)**	4	**ÿ2147483 648 to 2147483647**

Class of numeric types (Integer, Real, Complex)

- The type of real numbers is similar to the set of real numbers in mathematics

How to declare	Number of bytes occupied	Degree main body
REAL **REAL*4** **REAL (KIND=4)**	4	6
REAL*8 **REAL (KIND=8)** **DOUBLE PRECISION**	8	15

Class of number types (Integer, Real, Complex) •

A complex number is an ordered pair of two real numbers called the real part and the imaginary part

How to declare	Number of bytes occupied
COMPLEX **COMPLEX *4** **COMPLEX (4)** **COMPLEX (KIND=4)**	8
COMPLEX *8 **COMPLEX (8)** **COMPLEX (KIND=8)** **DOUBLE CPMPLEX**	16

Character type (Character)

• Character types have a set of values consisting of characters formed into strings

(strings. • The

length of the string is the number of characters in the declared string

newspaper.

• Each character in the character string occupies 1 byte of memory.

• **CHARACTER (length) vname**

Example: ***Character (10) Name***

Operations on data types •

Arithmetic operations: Used with integer , real and complex numbers.

- Relational operations, or comparison operations: Used with integers, real numbers, character types, and also with complex numbers in the case of equal or not equal comparisons. •

Logical operations: Used with logical types, and possibly with integers.

- Character string aggregation operation: Used with sig

on one's own.

Ký hiệu phép toán	Tên gọi/Ý nghĩa	Thứ tự ưu tiên	Thứ tự thực hiện	Ví dụ
		Phép toán số học		
**	Phép lũy thừa	1	Phải sang trái	A ** B
*	Phép nhân	2	Trái sang phải	A * B
/	Phép chia	2	Trái sang phải	A / B
+	Phép cộng	3	Trái sang phải	A + B
–	Phép trừ	3	Trái sang phải	A – B
		Phép toán quan hệ		
.EQ. (==)	Bằng	–	Không phân định	A.EQ.B; A == B
.LT. (<)	Nhỏ hơn	–	Không phân định	A.LT.B; A < B
.LE. (<=)	Nhỏ hơn hoặc bằng	–	Không phân định	A.LE.B; A <= B
.GT. (>)	Lớn hơn	–	Không phân định	A.GT.B; A > B
.GE. (>=)	Lớn hơn hoặc bằng	–	Không phân định	A.GE.B; A >= B
.NE. (/=)	Không bằng (Khác)	–	Không phân định	A.NE.B; A /= B
		Phép toán lôgic		
.NOT.	Phủ định	1	Không phân định	.NOT. L1
.AND.	Và (Phép hội)	2	Trái sang phải	L1. AND. L2
.OR.	Hoặc (Phép tuyển)	3	Trái sang phải	L1. OR. L2
.XOR.	Hoặc triệt tiêu	4	Trái sang phải	L1. XOR. L2
.EQV.	Tương đương	4	Trái sang phải	L1. EQV. L2
.NEQV.	Không tương đương	4	Trái sang phải	L1. NEQV. L2
		Gộp ký tự		
//	Gộp hai xâu ký tự	–	Trái sang phải	ST1 // ST2

- **PARAMETER:** *constant* property , •

DIMENSION: array property , •

ALLOCATABLE: *dynamically allocated* property , •

POINTER: *pointer* property

- ……

Some basic functions and procedures in Fortran

• **ABS(A)** Absolute value of integer, real or complex number A • **ACOS(X)** Arccosine (inverse function of cosine) of X • **AIMAG(Z)** Imaginary part of complex number Z • **ASIN(X)** function inverse of sine X • **ATAN** (X) **inverse** function of tangent hyperbola of X • **EXP(X) e** ^{x} • LOG (X) Natural logarithm of X • **LOG10(X)** Base 10 logarithm of smallest of numbers A1, A2, A3,… • **MIN(A1,A2[,A3,...])** Minimum value of numbers A1, A2, A3,… • **MOD(A, P)** Balance of division A by P

Input - output from the

screen Write (*,*) a: Write the number

a on the screen Write (*,*) „……ÿ: Write a prompt, comma
on-screen captions.

Read (*,*) a : Input the number a from the screen

Unconditional loop

Since I =1,n,m

.....

Enddo

I: Running index

N: Limit the running index

M: The jump of the runner

Program testing *Real c* *C=0.* *since 10 i=1.10* *C=C+i* *10 write(*,*) c*	*Program testing* *Real c* *C=0.* *since i=1.10* *C=C+i* *enddo* *write(*,*) c*	*Program testing* *Real c* *C=0.* *since 10 i=1.10* *C=C+i* *10 continued* *write(*,*) c*

Exercise

1) Calculate the factorial (given number and number entered from the screen)

2) Calculate the total (given number and number entered from the screen)

3) Enter and export the name from the screen

Practice (October 16, 2012)

PRACTICE
FORTRAN

Commands branch with if

```
IF (BThuc_Logic) THEN
The_commands
END IF
```

```
IF (BThuc_Logic) THEN
Commands_1
ELSE
Commands_2
END IF
```

```
IF (BThuc_Logic_1) THEN
Commands_1
ELSE IF (BThuc_Logic_2) THEN
Commands_2
ELSE IF (BThuc_Logic_3) THEN
Commands_3
...
ELSE
Commands_n
END IF
```

Import - export data from existing files

Open (unit=1,file=„data.txt',status='old')

Status = old, new, unknown

Unit = name of file referenced in program File =

'name of export, import file'

=> Receive ASCII files: *.dat, *.txt

Exercise

1) Find max, min.

2) Write a program to calculate the

average 3) Write a program to find the roots of any quadratic equation .

4) Write a program to sort from largest to smallest.

Data format The

Format command allows exporting and importing in a format requested by the

user. 10 format (“comment”, format)

Format:

Command Data type ***Iw.m*** Integer
type ***Fw.d*** Real number
type ***Aw*** Character
type w: Total number

of characters in the number

range d, m: Decimal part

Directions when programming

- It's easier to rewatch the program you've already seen write.

- It's easy for others to understand your programming.

- Make good use of written programs ÿ+ Write comments about programs, variables (commands!) ÿ+ Hierarchize commands by indentation ÿ+ Separate parts of the program ÿ+ Use capital and case letters usually reasonable ÿ+ Use spaces in correct syntax

The generation cycle is undetermined

If and goto

m *Statement_at the beginning_of_loop*
The_next_statements_in_the_loop

_

IF *(BThuc_Logic)* GOTO m

m *Statement_at the beginning_of_loop*
The_next_statements_in_the_loop

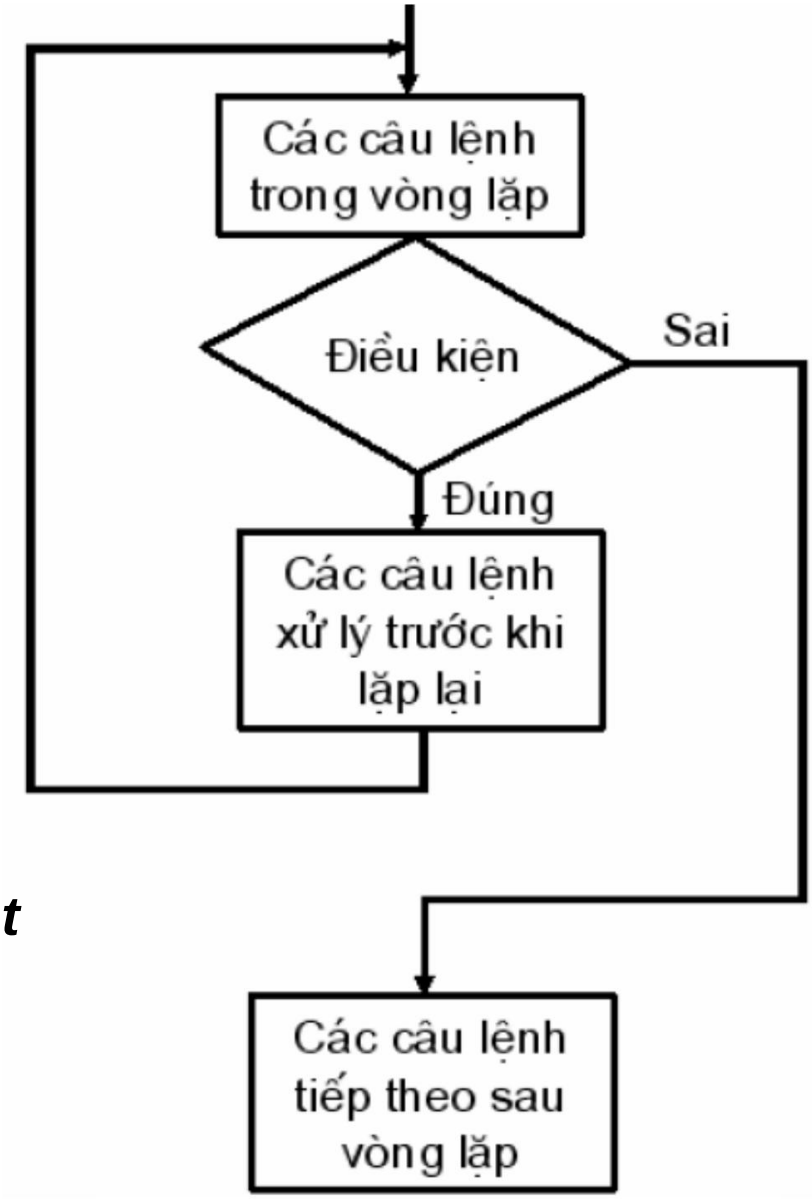
_

IF *(BThuc_Logic)* THEN
Processing_statements_before_repeat

_

GOTO m

END IF



- Recall what you have learned:

Unconditional loop

Since I =1,n,m

……..

Enddo

I: Running index

N: Limit the running index

M: The jump of the runner

DO WHILE…END DO

DO ***WHILE(BThuc_Logic)***

The_commands

END DO

(BThuc_Logic) => true => execute statements until (BThuc_Logic) => false

- **Exercises in class**

1. Enter any number from the keyboard, if the number is negative, ask to re-enter, if the number is positive, output out the screen.
2. Find the largest positive integer n that satisfies the inequality:

$$2n2 - 8n < 50$$

Suggestion 1:

```
Program dowhile1
Real*4 a
Write(*,*) 'Enter any value'
Read(*,*) a
Do while (a<0)
        Write(*,*) 'enter the value of a line'
        Read(*,*) a
Enddo
Write(*,*) a
End
```

Suggestion 2:

```
Program dowhile2
Integer*4 A, N
N=0
A= 2*N**2-8*N
Do while (A<50)
        N=N+1
         A= 2*N**2-8*N
Enddo
N=N-1
Write(*,*) N
End
```

Exercise

1) Dùng phương pháp chia đôi khoảng cách tính nghiệm của phương trình f(x)=x.sin(x)-1 trong khoảng [0,2]
2) Cho hàm số $e^{-x} - x = 0$, tìm nghiệm của phương trình bằng phương pháp lặp, với $x_0=0.5$ và $|x(i+1) - x(i)| \leq 0.00005$

Công thức lặp: $X(i+1) = e^{-x(i)}$

Output standard

Standard output	Description (Level of detail – action)	Level (I/T/U)
G1.2	Understand some internal commands and structures fortran	TU
G1.2	Apply the Fortran programming language in data processing and calculating simple	TU
G1.3	problems. Apply some drawing tools and present results.	T
G2	Generate ideas to solve problems	TU
G3	Determine the appropriate algorithm	TU
G4	Have independent and creative thinking	TU

ARRAY IN FORTRAN

- One-dimensional array
- Two-dimensional array
- Three-dimensional array

Array name

Number of dimensions

Maximum size

How to sort array elements

- A set of elements of the same data type arranged in a certain order
- Each element is identified by its index and value

- **Array declaration**

REAL A(10, 2, 3) ! *Array of three-dimensional real*

numbers **DIMENSION A(10, 2, 3) !** *3-dimensional array of real*

numbers **ALLOCATABLE B(:, :) !** *2-dimensional array of real numbers*

REAL,DIMENSION (2,5) :: D ! *2-dimensional array of real*

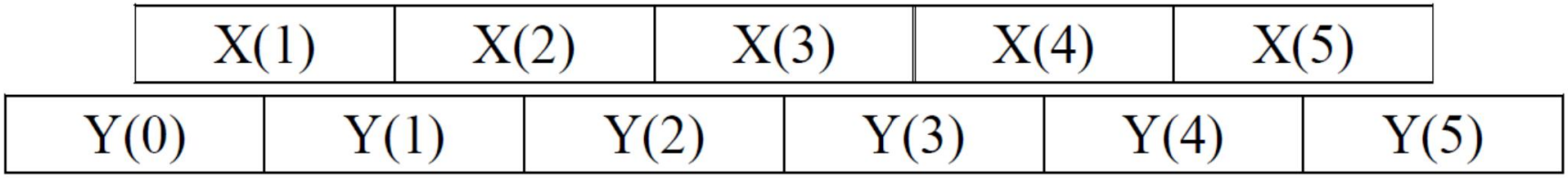
numbers **REAL, ALLOCATABLE :: E(:,:,:) !** *3-dimensional real ar*

INTEGER M(10, 10, 10)

INTEGER K(-3:6, 4:13, 0:9)

- **Store arrays in memory**

REAL X(5), Y(0:5)




```
REAL X (4, 7, 9)
...
CALL SUB1(X)
CALL SUB2(X)
...
END
SUBROUTINE SUB1(A)
REAL A(:, :, :)
...
END SUBROUTINE SUB1
SUBROUTINE SUB2(B)
REAL B(3:, 0:, -2:)
...
END SUBROUTINE SUB2
```

→ **A (4, 7, 9)**

→ **B (3:6, 0:6, -2:6)**

```
REAL X(5), Y(0:5)

Y(0) = 1.

DO I=1.5
        X(I) = I*I
        Y(I) = X(I) + I
END DO

PRINT '(6F7.1))', (X(I), I=1.5)
PRINT '(6F7.1))', (Y(I), I=0.5)

END
```

Static array and dynamic array

Static array

The memory area reserved for array storage is fixed and it is not freed as long as the program is active

The size of a static array cannot be changed during program execution

Dynamic array

The area of memory that stores it can be assigned, changed, and released while the program is executing

Initialize the array using the DATA command

```
REAL, DIMENSION(10) :: A, B, C(3,3)
DATA A / 5*0, 5*1 /
DATA B(1:5) / 4, 0, 5, 2, -1 /
DATA ((C(I,J), J= 1,3), I=1,3) /3*0,3*1, 3*2/
DATA C / 3*0, 3*1, 3*2 /
```

Array expression

REAL, DIMENSION(10) :: X, Y

X + Y ! X(I) + Y(I)

X * Y ! X(I) * Y(I)

X * 3 ! X(I) * 3

X * SQRT(Y) ! Y: X(I) * SQRT(Y(I))

X == Y

Structure WHERE... ELSEWHERE ... END WHERE

WHERE *(Condition)* Statement

```
WHERE (Condition)
      Commands_1
ELSEWHERE
      Commands_2
END WHERE
```

```
REAL A (5)
    A = (/ 89.5, 43.7, 126.4, 68.3, 137.7 /)
WHERE (A > 100.0) A = 100.0
```

```
REAL A (5)
A = (/ 89.5, 43.7, 126.4, 68.3, 137.7 /)
WHERE (A > 100.0) A = 100.0
```

A = (89.5, 43.7, 100.0, 68.3, 100.0)

SUBPROGRAM (SUBROUTINE AND FUNCTION)

- there are parts that are often used repeatedly many times
- program segments can be used for other programs • writing

a program with many duplicate segments will cause boredom, ineffectiveness, and even make the program more confusing

Classify

- There are two subprogram concepts:

+ Procedure (SUBROUTINE): returns a variable through the variable's argument

+ Function (FUNCTION): Function: Returns a value through the

function name • This set of subprograms is called a

module • The structure of the subprograms is the same as the main program

- Placed at the end of the main program

FUNCTION

```
PROGRAM VER2
INTEGER I
DO I = 1, 10
        WRITE(*,*) I, GT(I)
END DO


CONTAINS !------------------------------------- FUNCTION GT ( N )
END
FUNCTION END
statement
```

FUNCTION

```
PROGRAM VER2
INTEGER I
DO I = 1, 10
        WRITE(*,*) I, GT(I)
END DO


CONTAINS !---------------------------------------- FUNCTION GT ( N )
INTEGER GT, N, Tam, I
Tam =
1 DO I = 2,
        N Tam = I *
Tam
END DO
GT = Tam END
FUNCTION END
```

SUBROUTINE

- There is no value associated with the procedure name
- To call a procedure, the CALL keyword must

be used • The SUBROUTINE keyword is used to define a procedure instead of the FUNCTION

keyword • A function without arguments will be called by adding empty parentheses after the function name

```
program test
parameter (n=5)
real*4 a(n),tbc
open (1, file='data.txt', status='unknown')
do i=1,n
        read(1,*) a(i)
        write(*,*) i, a(i)
enddo
call tb(a1,n1,tbc1)
write(*,*) tbc
close(1)
contains
!------------------------------------------ -------------------------------
Subroutine tb(a,n,tbc)
        real*4 a(n) ,kq,tbc
kq=0.
since i=1,n
        Kq=a(i)+kq
enddo
        tbc=kq/n
endsubroutine
end
```

CONTAINS statement

- The unexecuted statement is used to separate the main program body from its *subprograms .* Internal subprograms are arranged immediately after the **CONTAINS** statement and before the **END** keyword of the main program

GLOBAL VARIABLES

LOCAL VARIABLES

```
PROGRAM VER2
INTEGER I
DO I = 1, 10
        WRITE(*,*) I, GT(I)
END DO

CONTAINS !----------------------------------------
INTEGER GT, N, Tam, I
Tam =
1 DO I = 1,
        N Tam = I *
Tam
END DO
GT = Tam END
FUNCTION END
```

```
PROGRAM VER1
INTEGER I
DO I = 1, 10
        WRITE(*,*) I, GT(I)
END DO

CONTAINS !----------------------------------------
INTEGER Gt, N, Tam
Tam =
1 DO I = 1,
        N Tam = I *
Tam
END DO
GT = Tam END
FUNCTION END
```

- Variable **I** declared in the main program is called a *global variable,* while variable **I** declared in a

subprogram is *a local variable.* • Internal subprograms *are allowed* to refer to *global variables* when

local variables are not declared. • However, the main program will *not be able to refer* to *local variables* declar

Request:

1) Master how to properly open and close workspaces, avoid running multiple programs on top of each other.

2) Understand the structure of writing a fortran program.

3) Proficient in declaring and using various types of variables.

4) Ways to export, import, and assign variables to values in Fortran (directly from the screen and from the file).

5) Proficient in using Do and If loops

Types of practice exercises:

1) Write a simple calculation program: sum,
product, factorial, find min, max, calculate average...

2) Practice for If loop: solve quadratic equations,
if….else….., if…..elseif…..

3) Allow input and output values from the keyboard and
available files, process and calculate on that data.

4) Some other advanced exercises (if any)

Lesson 1: Write a subprogram to calculate cos(x) according to recipe:

$$\cos(x) \approx 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots\cdots$$

With accuracy ÿ= 0.0001

Use subroutine to calculate factorial

Lesson 2: Write a subprogram to solve the equation

$$Ax^2 + Bx + C = 0$$

Level 2: , apply the calculation with an example for any A, B, C entered from the keyboard. If The equation has no solution, assign X1=X2=-999.

Grading scale for practice exercises 1.

Fully and accurately declare all variables used in the program: 2 points, minus 0.25 points for each missing or incorrectly declared variable. If more than 6 violating variables will no longer have points in this section.

2. Open the file, read or write data to the file as required: 1 point, no points for errors.

3. Present the parts of the program in the correct order : 1 pt. Any part that is not arranged correctly will be deducted 0.5 points

4. Content: scoring according to the results achieved, the correct results with full points; Correct idea: ¼ points, correct command usage: ¼ half points, correct suggestions and corrections: ¼ points .

- https://sites.google.com/site/nthanhbanvl/tin-study-lop-11-1/thuc-hanh

Đại Học Khoa Học Tự Nhiên Tp.HCM
Khoa Vật Lý – Vật Lý Kỹ Thuật
Bộ Môn Hải Dương, Khí Tượng và Thủy Văn

LẬP TRÌNH ỨNG DỤNG

ThS. Nguyễn Hoàng Phong
ThS. Nguyễn Tiến Thành

Mục tiêu của môn học

- Áp dụng ngôn ngữ lập trình trong tính toán
- Hình thành ý tưởng giải quyết bài toán
- Xác định thuật toán phù hợp
- Có tư duy độc lập, sáng tạo

Chuẩn đầu ra của môn học

- Hiểu được một số lệnh và cấu trúc trong fortran
- Áp dụng ngôn ngữ lập trình fortran trong sử lý số liệu và tính toán các bài toán đơn giản
- Áp dụng một số công cụ vẽ hình, trình bày kết quả
- Hình thành ý tưởng giải quyết bài toán
- Xác định thuật toán phù hợp
- Có tư duy độc lập, sáng tạo

Nội dung môn học

I. Phần mềm hỗ trợ

- Surfer
- Grapher
- WRPLOT view

II. Fortran căn bản

- Ngôn ngữ lập trình fortran
- Numerical recipes

III. Toolbox trong matlap

Phân bố thời gian

1) Giới thiệu chung
2) Ngôn ngữ lập trình fortran
3) Ngôn ngữ lập trình fortran (tt)
4) Thực hành ngôn ngữ lập trình fortran
5) Thực hành ngôn ngữ lập trình fortran (tt)
6) Ứng dụng lập trình fortran giải bài toán (gk)
7) Ứng dụng lập trình fortran giải bài toán (tt)
8) Numerical recipes fortran

Phân bố thời gian (tt)

9) Thực hành Sufer

10) Thực hành grapher, WRPLOT view (bt)

11) Ôn tập matlap

12) Toolbox trong matlap

13) Ứng dụng toolbox trong thực tế (bt)

14) Ôn tập

15) Ôn tập (tt)

Tài liệu

- Phan Văn Tân (2007), Ngôn ngữ lập trình Fortran 90, NXB Đại học Quốc gia Hà Nội.
- Fortran 90 User's Guide
- Brian D Hahn (1994), FORTRAN 90 for Scientists and Engineers, Butterworth-Heinemann

Quy định

- Sinh viên cần tuân thủ nghiêm túc các nội quy và quy định của Khoa và Trường.
- Sinh viên không được vắng quá 3 buổi trên tổng số các buổi học lý thuyết, vắng quá số buổi sẽ bị cấm thi.
- Sinh viên đi học đúng giờ, vào trễ không điểm danh.
- Đối với bất kỳ sự gian lận nào trong quá trình làm bài tập hay bài thi, sinh viên phải chịu mọi hình thức kỷ luật của Khoa/Trường và bị 0 điểm cho môn học này.

Cơ cấu điểm

- Thi cuối kỳ: 50% (LT (mở): 30%, TH: 20%)
- Thi giữa kỳ: 20%
- Bài tập nhỏ: 20%
- Chuyên cần: 10%

MỞ ĐẦU FORTRAN

Giới thiệu

1) Ngôn ngữ lập trình: là một hệ thống được ký hiệu hóa để miêu tả những tính toán (qua máy tính) trong một dạng mà cả con người và máy đều có thể đọc và hiểu được.

- Dễ hiểu và dễ sử dụng đối với người lập trình, để có thể dùng để giải quyết nhiều bài toán khác nhau.

- Miêu tả một cách đầy đủ và rõ ràng các tiến trình (*process*), để chạy được trên các hệ máy tính khác nhau.

Mỗi ngôn ngữ lập trình cũng chính là một chương trình, nhưng nó có thể được dùng để tạo nên các chương trình khác. Văn bản được viết bằng ngôn ngữ lập trình để tạo nên chương trình được gọi là mã nguồn.

Ngôn ngữ lập trình là một tập con của ngôn ngữ máy tính, được thiết kế và chuẩn hóa để truyền các chỉ thị cho các máy có bộ xử lý (CPU), nói riêng là máy tính. Ngôn ngữ lập trình được dùng để lập trình máy tính, tạo ra các chương trình máy nhằm mục đích điều khiển máy tính hoặc mô tả các thuật toán để người khác đọc hiểu.

Ngôn ngữ máy (machine language) là các chỉ thị dưới dạng nhị phân, can thiệp trực tiếp vào trong các mạch điện tử.

- Chương trình được viết bằng ngôn ngữ máy thì có thể được thực hiện ngay không cần qua bước trung gian nào.
- Tuy nhiên chương trình viết bằng ngôn ngữ máy dễ sai sót, cồng kềnh và khó đọc, khó hiểu vì toàn những con số 0 và 1.

Hợp ngữ (assembly language) là một bước tiến vượt bậc đưa ngôn ngữ lập trình thoát ra khỏi ngôn ngữ máy khó hiểu.Xuất hiện vào những năm 1950, nó được thiết kế để máy tính trở nên thân thiện hơn với người sử dụng.

- Hợp ngữ đưa ra khái niệm biến (variable).

- Hợp ngữ cũng chứa vài "phép toán giả", có thể biểu biễn mã phép toán dưới dạng phát biểu (hay còn gọi là câu lệnh) thay vì dưới dạng nhị phân. Các câu lệnh bao gồm hai phần: phần mã lệnh (viết tựa tiếng Anh) chỉ phép toán cần thực hiện và phần tên biến chỉ địa chỉ chứa toán hạng của phép toán đó.

- Để máy thực hiện được một chương trình viết bằng hợp ngữ thì chương trình đó phải được dịch sang ngôn ngữ máy. Công cụ thực hiện việc dịch đó được gọi là Assembler

Ngôn ngữ cấp cao (High level language) là ngôn ngữ được tạo ra và phát triển nhằm phản ánh cách thức người lập trình nghĩ và làm.

- Ngôn ngữ cấp cao rất gần với ngôn ngữ con người (Anh ngữ) nhưng chính xác như ngôn ngữ toán học. Nhờ ngôn ngữ cấp cao mà lĩnh vực lập trình trở nên phổ biến, rất nhiều người có thể viết được chương trình, và nhờ thế mà các phần mềm phát triển như vũ bão, phục vụ nhiều lĩnh vực của cuộc sống.

- Một chương trình viết bằng ngôn ngữ cấp cao được gọi là chương trình nguồn (source programs). Để máy tính "hiểu" và thực hiện được các lệnh trong chương trình nguồn thì phải có một chương trình dịch để dịch chương trình nguồn (viết bằng ngôn ngữ cấp cao) thành chương trình đích.

Chương trình viết trên ngôn ngữ bậc cao phải sử dụng một chương trình dịch để chuyển đổi thành chương trình trên ngôn ngữ máy mới có thể thực hiện.

Lập trình bằng ngôn ngữ bậc cao dễ viết hơn vì các lệnh được mã hóa gần với ngôn ngữ tự nhiên.

Lập trình trên ngôn ngữ máy rất khó, thường các chuyên gia lập trình mới lập trình được.

2) Lập trình (computer programming, thường gọi tắt là programming) là việc sử dụng cấu trúc dữ liệu và các lệnh của ngôn ngữ lập trình cụ thể để mô tả dữ liệu và diễn đạt các thao tác của thuật toán.

3) Các thành phần của ngôn ngữ lập trình

Bảng chữ cái →

-Tập các kí hiệu dùng để viết chương trình
-Từ A-Z

Cú pháp →

- Bộ qui tắc dùng để viết chương trình
- Cách viết một chương trình hợp lệ
- **Write (*,*) a**

Ngữ nghĩa →

- Xác định ý nghĩa của các tổ hợp kí tự trong chương trình

4. Một số khái niệm

4.1 Tên: Mọi đối tượng trong chương trình đều phải được đặt tên theo quy tắc của ngôn ngữ lập trình và từng chương trình dịch cụ thể

- **Tên dành riêng:** Là những tên được ngôn ngữ lập trình quy định với ý nghĩa xác định

Mà người lập trình không thể dùng với ý nghĩa khác. Tên dành riêng khi viết đúng sẽ hiện màu xanh.

- **Tên do người lập trình tự đặt**: Được xác định bằng cách khai báo trước khi sử dụng và không được trùng với tên dành riêng; Các tên trong chương trình không được trùng nhau

4.2 Hằng và biến

Hằng: Là các đại lượng có giá trị không đổi trong quá trình thực hiên chương trình

Biến: Là đại lượng được đặt tên, giá trị có thể thay đổi được trong chương trình

- **Sơ lược lịch sử về ngôn ngữ lập trình fortran**

- Tập hợp các qui tắc đặc biệt để mã hoá những kiến thức cho máy tính hiểu được gọi là ngôn ngữ lập trình.
- John Backus đề xuất vào khoảng cuối năm1953 ở New York ⇒ tháng 4 năm 1957.
- 1966, phiên bản chuẩn đầu tiên.
- Fortran 77 (1978) ⇒ Fortran 90 (1991) ⇒ Fortran 95, Fortran 2003, Fortran 4.0, Fortran 6.0,….
- Fortran có thể làm việc trên nhiều hệ điều hành khác nhau, như các dòng UNIX, LINUX, WINDOWS, DOS,…

Cấu trúc 1 chương trình fortran cơ bản

```
Program tinh_do_cao_song
integer*4 n, d
Parameter (n=2000,d=23)
integer*4 i,j,m,k
real*4 a(n),b(n),c(n),h(n),tam,e(d),f(d),x(d)
Open (unit=1,file='KQKT_cs_00_0902.txt',status='old')
----------------------------------------
Close (1)
End
Subroutine tb(a,n,tbc)
----------------------------------------
Endsubroutine
```

- Gõ lời chương trình (*source code*)

- Chạy (*run*)

- Biên dịch (*compile*): dịch (*translated*) sang mã máy (*machine code)*

+ Kiểm tra về cú pháp (*Syntax*)

+Dịch sang mã máy tính

+ Đối tượng (*Object*)

+ Liên kết (*Link*)

+ File *có thể thực hiện* (*executable*)

- Thực hiện (*executed*)

Cách đặt tên biến và tên hằng

- Có thể gồm 1 đến 31 ký tự
- Phải bắt đầu bởi một chữ cái tiếng Anh. Các ký tự được sử dụng để cấu tạo tên biến, tên hằng gồm 26 chữ cái tiếng Anh
- Không phân biệt chữ thường, chữ hoa, (A−Z và a−z), 10 chữ số (0−9), và *dấu gạch dưới* (_)

VD:

~~a) 2A~~

~~b) An An~~

c) delta_4x

~~d) theta+alpha~~

Khai báo kiểu dữ liệu cho biến

- 5 kiểu dữ liệu chuẩn  hai lớp

+ Lớp các kiểu số (*numeric)*

- Số nguyên (**integer**)
- Số thực (**real**)
- Số phức (**complex**)

+ Lớp các kiểu không phải số (*nonnumeric)*

- Kiểu ký tự (**character**)
- Kiểu lôgic (**logical**)

Lớp các kiểu số (Integer, Real, Complex)

- Kiểu số nguyên là những dữ liệu nhận các giá trị thuộc tập hợp số nguyên

Cách khai báo	Số byte chiếm giữ	Khoảng giá trị
INTEGER	4	**−2147483 648 đến 2147483 647**
INTEGER*1 hoặc **INTEGER (1)** hoặc **INTEGER (KIND=1)**	1	**−128 đến 127**
INTEGER*2 hoặc **INTEGER (2)** hoặc **INTEGER (KIND=2)**	2	**−32 768 đến 32 767**
INTEGER*4 hoặc **INTEGER (4)** hoặc **INTEGER (KIND=4)**	4	**−2147483 648 đến 2147483647**

Lớp các kiểu số (Integer, Real, Complex)

- Kiểu số thực gần giống với tập số thực trong toán học

Cách khai báo	Số byte chiếm giữ	Độ chính xác
REAL **REAL*4** **REAL (KIND=4)**	4	6
REAL*8 **REAL (KIND=8)** **DOUBLE PRECISION**	8	15

Lớp các kiểu số (Integer, Real, Complex)

- Số phức là một cặp có thứ tự của hai số thực được gọi là phần thực và phần ảo

Cách khai báo	Số byte chiếm giữ
COMPLEX **COMPLEX *4** **COMPLEX (4)** **COMPLEX (KIND=4)**	8
COMPLEX *8 **COMPLEX (8)** **COMPLEX (KIND=8)** **DOUBLE CPMPLEX**	16

Kiểu ký tự (Character)

- Kiểu ký tự có tập giá trị là các ký tự lập thành xâu (chuỗi) ký tự.
- Độ dài của xâu là số ký tự trong xâu đã được khai báo.
- Mỗi ký tự trong xâu ký tự chiếm 1 byte bộ nhớ.
- **CHARACTER (length) vname**

VD: ***Character (10) Name***

Phép toán trên các kiểu dữ liệu

- Phép toán số học: Sử dụng với các kiểu số nguyên, số thực và số phức.
- Phép toán quan hệ, hay phép toán so sánh: Sử dụng với các kiểu số nguyên, số thực, kiểu ký tự, và cũng có thể đối với cả số phức trong trường hợp so sánh bằng hoặc không bằng.
- Phép toán lôgic: Sử dụng với kiểu lôgic, và có thể với cả số nguyên.
- Phép toán gộp xâu ký tự: Sử dụng với kiểu ký tự.

Ký hiệu phép toán	Tên gọi/Ý nghĩa	Thứ tự ưu tiên	Thứ tự thực hiện	Ví dụ
Phép toán số học				
**	Phép lũy thừa	1	Phải sang trái	A ** B
*	Phép nhân	2	Trái sang phải	A * B
/	Phép chia	2	Trái sang phải	A / B
+	Phép cộng	3	Trái sang phải	A + B
–	Phép trừ	3	Trái sang phải	A – B
Phép toán quan hệ				
.EQ. (==)	Bằng	–	Không phân định	A.EQ.B; A == B
.LT. (<)	Nhỏ hơn	–	Không phân định	A.LT.B; A < B
.LE. (<=)	Nhỏ hơn hoặc bằng	–	Không phân định	A.LE.B; A <= B
.GT. (>)	Lớn hơn	–	Không phân định	A.GT.B; A > B
.GE. (>=)	Lớn hơn hoặc bằng	–	Không phân định	A.GE.B; A >= B
.NE. (/=)	Không bằng (Khác)	–	Không phân định	A.NE.B; A /= B
Phép toán lôgic				
.NOT.	Phủ định	1	Không phân định	.NOT. L1
.AND.	Và (Phép hội)	2	Trái sang phải	L1. AND. L2
.OR.	Hoặc (Phép tuyển)	3	Trái sang phải	L1. OR. L2
.XOR.	Hoặc triệt tiêu	4	Trái sang phải	L1. XOR. L2
.EQV.	Tương đương	4	Trái sang phải	L1. EQV. L2
.NEQV.	Không tương đương	4	Trái sang phải	L1. NEQV. L2
Gộp ký tự				
//	Gộp hai xâu ký tự	–	Trái sang phải	ST1 // ST2

- **PARAMETER:** thuộc tính *hằng*,
- **DIMENSION:** thuộc tính *mảng*,
- **ALLOCATABLE:** thuộc tính *cấp phát động*,
- **POINTER:** thuộc tính *con trỏ*
- ……

Một số hàm và thủ tục cơ bản trong fortran

- **ABS(A)** Giá trị tuyệt đối của số nguyên, số thực hoặc số phức A
- **ACOS(X)** Arccosine (hàm ngược của cosine) của X
- **AIMAG(Z)** Phần ảo của số phức Z
- **ASIN(X)** hàm ngược của sine X
- **ATAN(X)** hàm ngược của tang X, trong phạm vi $-\pi/2$ đến $\pi/2$
- **CONJG(Z)** Liên hợp phức của Z
- **COS(X)** Cosine của X
- **COSH(X)** Cosine hyperbol của X
- **EXP(X)** e^x
- **LOG(X)** Lôgarit cơ số tự nhiên của X
- **LOG10(X)** Lôgarit cơ số 10 của X
- **MAX(A1,A2[,A3,...])** Giá trị lớn nhất của các số A1, A2, A3,…
- **MIN(A1,A2[,A3,...])** Giá trị nhỏ nhất của các số A1, A2, A3,…
- **MOD(A, P)** Số dư của phép chia A cho P

Nhập - xuất từ màn hình

Write (*,*) a : Viết số a lên màn hình

Write (*,*) '……': Viết một câu lệnh nhắc, câu chú thích trên màn hình.

Read (*,*) a : Nhập vào số a từ màn hình

Vòng lặp không điều kiện

Do I =1,n,m

……..

Enddo

I : Chỉ số chạy

N: Giới hạn chỉ số chạy

M: Bước nhảy của chị số chạy

```
Program test
    Real c
C=0.
do 10 i=1,10
   C=C+i
10 write(*,*) c
```

```
Program test
    Real c
C=0.
do i=1,10
   C=C+i
enddo
write(*,*) c
```

```
Program test
    Real c
C=0.
do 10  i=1,10
   C=C+i
10 continue
write(*,*) c
```

Bài tập

1) Tính giai thừa (số cho sẵn và số nhập từ màn hình)
2) Tính tổng (số cho sẵn và số nhập từ màn hình)
3) Nhập xuất tên từ màn hình

Thực hành (16/10/12)

THỰC HÀNH FORTRAN

Lệnh rẽ nhánh với if

```
IF (BThuc_Logic) THEN
Các_câu_lệnh
END IF
```

```
IF (BThuc_Logic) THEN
Các_câu_lệnh_1
ELSE
Các_câu_lệnh_2
END IF
```

```
IF (BThuc_Logic_1) THEN
Các_câu_lệnh_1
ELSE IF (BThuc_Logic_2) THEN
Các_câu_lệnh_2
ELSE IF (BThuc_Logic_3) THEN
Các_câu_lệnh_3
...
ELSE
Các_câu_lệnh_n
END IF
```

Nhập – xuất dữ liệu từ file có sẵn

Open (unit=1,file=‘data.txt',status='old')

Status = old, new, unknown

Unit = tên file được tham chiếu trong chương trình

File = ‘tên file xuất, nhập’

=> Nhận file dạng ASCII: *.dat, *.txt

Bài tập

1) Tìm max, min.
2) Viết chương trình tính trung bình
3) Viết chương trình tìm nghiệm phương trình bậc 2 bất kỳ.
4) Viết chương trình sắp xếp từ lớn đến nhỏ.

Định dạng số liệu

Lệnh Format cho phép xuất, nhập theo một định dạng mà người sử dụng yêu cầu.

10 format ("chú thích", định dạng)

Định dạng:

Lệnh	Dạng dữ liệu
I*w.m*	Kiểu số nguyên
F*w.d*	Kiểu số thực
A*w*	Kiểu ký tự

w: Tổng số ký tự của dãy số

d, m: Phần thập phân

Định hướng khi lập trình

- Dễ dàng hơn khi xem lại chương trình mình đã viết.
- Người khác dễ hiểu phần lập trình của mình.
- Khai thác tốt các chương trình đã viết

$\Rightarrow$+ Viết chú thích về chương trình, biến (lệnh !)

$\Rightarrow$+ Phân cấp câu lệnh bằng cách thụt đầu dòng

$\Rightarrow$+ Phân tách các phần của chương trình

$\Rightarrow$+ Dùng chữ hoa và chữ thường hợp lý

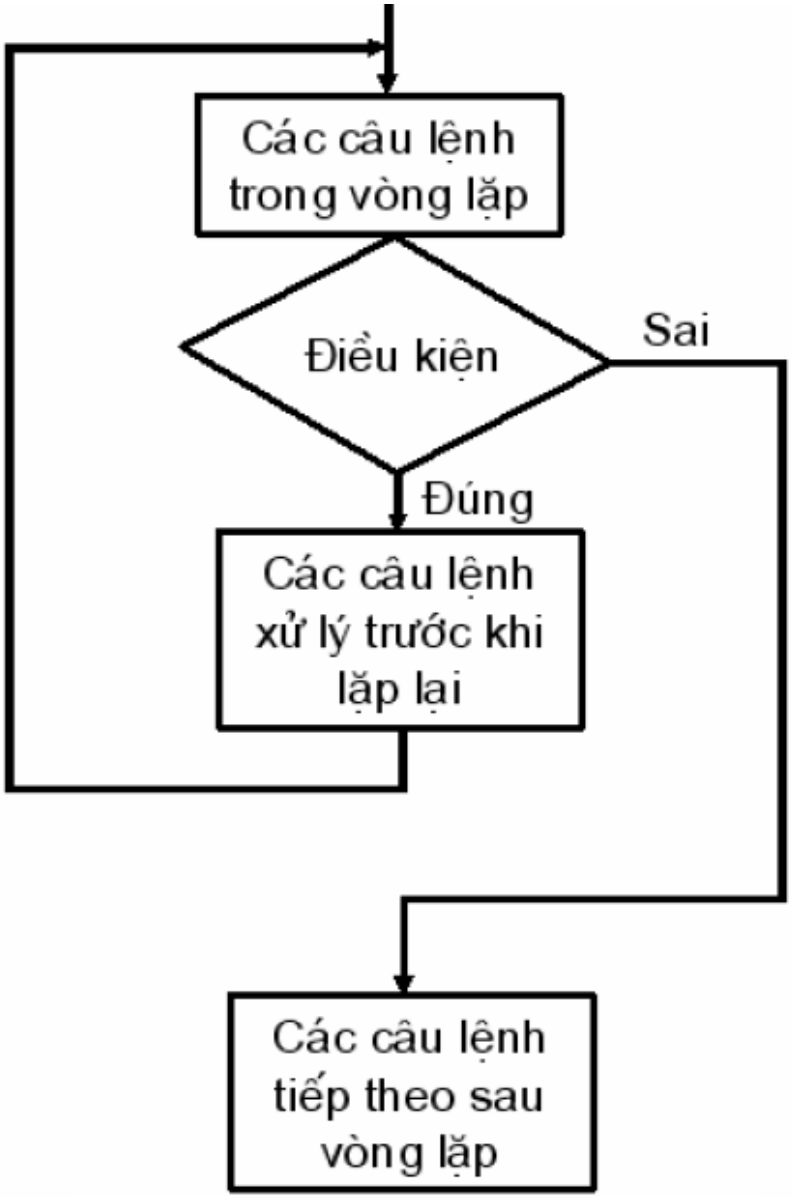
$\Rightarrow$+ Sử dụng dấu cách đúng cú pháp

Chu trình lập không xác định

If và goto

m ***Câu_lệnh_đầu_vòng_lặp***
Các_câu_lệnh_tiếp_theo_trong_vòng_lặp
IF (***BThuc_Logic***) **GOTO m**

m ***Câu_lệnh_đầu_vòng_lặp***
Các_câu_lệnh_tiếp_theo_trong_vòng_lặp
IF (***BThuc_Logic***) **THEN**
Các_câu_lệnh_xử_lý_trước_khi_lặp_lại
GOTO m
END IF



- Nhắc lại kiến thức đã học:

Vòng lặp không điều kiện

Do I =1,n,m

……..

Enddo

I : Chỉ số chạy

N: Giới hạn chỉ số chạy

M: Bước nhảy của chị số chạy

DO WHILE…END DO

DO WHILE(*BThuc_Logic*)
Các_câu_lệnh
END DO

(BThuc_Logic) => đúng => thực hiện các câu lệnh cho đến khi (BThuc_Logic) =>sai

- **Bài tập tại lớp**

1. Nhập vào 1 số bất kỳ từ bàn phím, nếu số đó âm, yêu cầu nhập lại, nếu số đó dương, xuất ra màn hình.
2. Tìm số nguyên dương n lớn nhất thỏa mãn bất phương trình:

$$2n^2-8n<50$$

Gợi ý 1:

```
Program dowhile1
Real*4 a
Write(*,*) 'Nhap gia tri bat ky'
Read(*,*) a
Do while (a<0)
        Write(*,*) 'nhap lai gia tri a duong'
        Read(*,*) a
Enddo
Write(*,*) a
End
```

Gợi ý 2:

```
Program dowhile2
Integer*4 A, N
N=0
A= 2*N**2-8*N
Do while (A<50)
        N=N+1
         A= 2*N**2-8*N
Enddo
N=N-1
Write(*,*) N
End
```

Bài tập

1) Dùng phương pháp chia đôi khoảng cách tính nghiệm của phương trình f(x)=x.sin(x)-1 trong khoảng [0,2]
2) Cho hàm số $e^{-x} - x = 0$, tìm nghiệm của phương trình bằng phương pháp lặp, với $x_0=0.5$ và $|\boldsymbol{x(i+1)} - \boldsymbol{x(i)}| \leq \mathbf{0.00005}$

Công thức lặp: $X(i+1) = e^{-x(i)}$

Chuẩn đầu ra

Chuẩn đầu ra	Mô tả (Mức chi tiết – hành động)	Mức độ (I/T/U)
G1.2	Hiểu được một số lệnh và cấu trúc trong fortran	TU
G1.2	Áp dụng ngôn ngữ lập trình fortran trong sử lý số liệu và tính toán các bài toán đơn giản	TU
G1.3	Áp dụng một số công cụ vẽ hình, trình bày kết quả	T
G2	Hình thành ý tưởng giải quyết bài toán	TU
G3	Xác định thuật toán phù hợp	TU
G4	Có tư duy độc lập, sáng tạo	TU

MẢNG TRONG FORTRAN

- Mảng một chiều
- Mảng hai chiều
- Mảng ba chiều

Tên mảng
Số chiều
Kích thước cực đại
Cách sắp xếp các phần tử của mảng

→ Một tập hợp các phần tử có cùng kiểu dữ liệu

→ Được sắp xếp theo một trật tự nhất định

→ Mỗi phần tử được xác định bởi chỉ số và giá trị của chúng

- **Khai báo mảng**

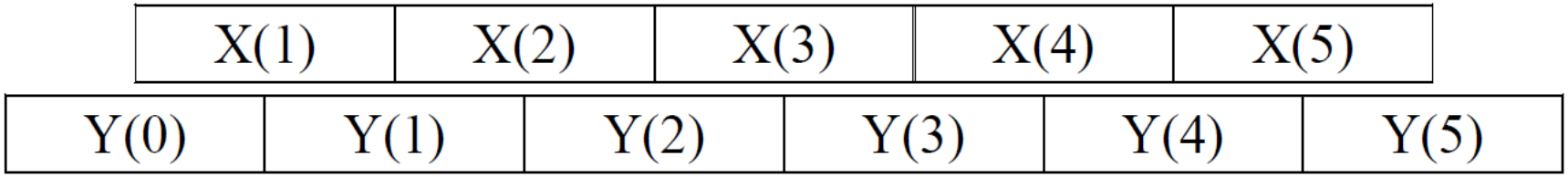
REAL A(10, 2, 3) *! Mảng các số thực 3 chiều*
DIMENSION A(10, 2, 3) *! Mảng các số thực 3 chiều*
ALLOCATABLE B(:, :) *! Mảng các số thực 2 chiều*

REAL,DIMENSION (2,5) :: D *! Mảng các số thực 2 chiều*
REAL, ALLOCATABLE :: E(:,:,:) *! Mảng thực 3 chiều*

INTEGER M(10, 10, 10)
INTEGER K(-3:6, 4:13, 0:9)

- **Lưu trữ mảng trong bộ nhớ**

REAL X(5), Y(0:5)




```
REAL X (4, 7, 9)
...
CALL SUB1(X)
CALL SUB2(X)
...
END
SUBROUTINE SUB1(A)
REAL A(:, :, :)
...
END SUBROUTINE SUB1
SUBROUTINE SUB2(B)
REAL B(3:, 0:, -2:)
...
END SUBROUTINE SUB2
```

→ **A (4, 7, 9)**

→ **B (3:6, 0:6, -2:6)**

```
REAL X(5), Y(0:5)

Y(0) = 1.

DO I=1,5
        X(I) = I*I
        Y(I) = X(I) + I
END DO

PRINT '(6F7.1))', (X(I), I=1,5)
PRINT '(6F7.1))', (Y(I), I=0,5)

END
```

Mảng tĩnh và mảng động

Mảng tĩnh

Vùng bộ nhớ dành lưu trữ mảng là cố định và nó không bị giải phóng chừng nào chương trình còn hiệu lực

Kích thước của mảng tĩnh không thể bị thay đổi trong quá trình chạy chương trình

Mảng động

Vùng bộ nhớ lưu trữ nó có thể được gán, thay đổi và giải phóng khi chương trình đang thực hiện

Khởi tạo mảng bằng lệnh DATA

```
REAL, DIMENSION(10) :: A, B, C(3,3)
DATA A / 5*0, 5*1 /
DATA B(1:5) / 4, 0, 5, 2, -1 /
DATA ((C(I,J), J= 1,3), I=1,3) /3*0,3*1, 3*2/
DATA C / 3*0, 3*1, 3*2 /
```

Biểu thức mảng

REAL, DIMENSION(10) :: X, Y
X + Y ! X(I) + Y(I)
X * Y ! X(I) * Y(I)
X * 3 ! X(I) * 3
X * SQRT(Y) ! Y: X(I) * SQRT(Y(I))
X == Y

Cấu trúc WHERE... ELSEWHERE ... END WHERE

WHERE (*Điều_kiện*) Câu_lệnh

WHERE *(Điều_kiện)*
Các_câu_lệnh_1
ELSEWHERE
Các_câu_lệnh_2
END WHERE

REAL A (5)
A = (/ 89.5, 43.7, 126.4, 68.3, 137.7 /)
WHERE (A > 100.0) A = 100.0

```
REAL A (5)
A = (/ 89.5, 43.7, 126.4, 68.3, 137.7 /)
WHERE (A > 100.0) A = 100.0

A = (89.5, 43.7, 100.0, 68.3, 100.0)
```

CHƯƠNG TRÌNH CON (SUBROUTINE VÀ FUNCTION)

- có những bộ phận thường được sử dụng lặp đi lặp lại nhiều lần
- những đoạn chương trình có thể được sử dụng cho các chương trình khác
- viết một chương trình trong đó có nhiều đoạn trùng lặp sẽ gây ra sự nhàm chán, không hiệu quả, thậm chí làm cho chương trình trở nên rối rắm hơn

Phân loại

- Có hai khái niệm chương trình con là:

+ Thủ tục (SUBROUTINE): trả về 1 biến thông qua đối số của biến

+ Hàm (FUNCTION): Function: Trả về 1 giá trị thông qua tên hàm

- Tập hợp các chương trình con này được gọi là modul
- Cấu trúc của các chương trình con giống như chương trình chính
- Được đặt ở cuối chương trình chính

FUNCTION

```
PROGRAM VER2
INTEGER I
DO I = 1, 10
        WRITE(*,*) I, GT(I)
END DO
CONTAINS
!-------------------------------------
FUNCTION GT ( N )
Câu lệnh
END FUNCTION
END
```

FUNCTION

```
PROGRAM VER2
INTEGER I
DO I = 1, 10
        WRITE(*,*) I, GT(I)
END DO
CONTAINS
!-------------------------------------
FUNCTION GT ( N )
INTEGER GT, N, Tam, I
Tam = 1
DO I = 2, N
        Tam = I * Tam
END DO
GT = Tam
END FUNCTION
END
```

SUBROUTINE

- Không có giá trị nào được liên kết với tên thủ tục
- Để gọi tới thủ tục phải dùng từ khóa CALL
- Từ khóa SUBROUTINE được dùng để định nghĩa thủ tục thay cho từ khóa FUNCTION
- Hàm không có đối số sẽ được gọi tới bằng cách thêm vào sau tên hàm cặp dấu ngoặc đơn rỗng

```
program test
parameter (n=5)
real*4 a(n),tbc
open (1, file='data.txt', status='unknown')
do i=1,n
        read(1,*) a(i)
        write(*,*) i, a(i)
enddo
call tb(a1,n1,tbc1)
write(*,*) tbc
close(1)
contains
!-------------------------------------------------------------------------------
Subroutine tb(a,n,tbc)
        real*4 a(n),kq,tbc
kq=0.
do i=1,n
        Kq=a(i)+kq
enddo
        tbc=kq/n
endsubroutine
end
```

Câu lệnh CONTAINS

- Câu lệnh không thực hiện, dùng để phân cách thân chương trình chính với các *chương trình con trong* thuộc nó. Các chương trình con trong được sắp xếp ngay sau câu lệnh **CONTAINS** và trước từ khóa **END** của chương trình chính

BIẾN TOÀN CỤC
BIẾN ĐỊA PHƯƠNG

```
PROGRAM VER2
INTEGER I
DO I = 1, 10
        WRITE(*,*) I, GT(I)
END DO
CONTAINS
!-------------------------------------
FUNCTION GT ( N )
INTEGER GT, N, Tam, I
Tam = 1
DO I = 1, N
        Tam = I * Tam
END DO
GT = Tam
END FUNCTION
END
```

```
PROGRAM VER1
INTEGER I
DO I = 1, 10
        WRITE(*,*) I, GT(I)
END DO
CONTAINS
!-----------------------------
FUNCTION GT ( N )
INTEGER Gt, N, Tam
Tam = 1
DO I = 1, N
        Tam = I * Tam
END DO
GT = Tam
END FUNCTION
END
```

- Biến **I** khai báo trong chương trình chính được gọi là *biến toàn cục*, còn biến **I** khai báo trong chương trình con là *biến địa phương*.
- Các chương trình con trong *được phép* tham chiếu đến các *biến toàn cục* khi các biến địa phương không được khai báo.
- Tuy nhiên, chương trình chính sẽ *không tham chiếu được* đến các *biến địa phương* khai báo ở các chương trình con trong.

Yêu cầu:

1) Nắm vững cách đóng mở và tạo workspace đúng cách, tránh việc chạy chồng nhiều chương trình lên nhau.
2) Hiểu cấu trúc viết 1 chương trình fortran.
3) Thành thạo khai báo, sử dụng các loại biến.
4) Các cách xuất, nhập, gán biến cho giá trị trong fortran (trực tiếp từ màn hình và từ file).
5) Sử dụng thành thạo vòng lặp Do, If

Các dạng bài tập thực hành:

1) Viết chương trình tính toán đơn giản: tổng, tích, giai thừa, tìm min, max, tính trung bình…
2) Thực hành cho vòng lặp If: giải phương trình bậc 2, if….else….., if…..elseif…..
3) Cho nhập, xuất giá trị từ bàn phím và file có sẵn, xử lý, tính toán trên số liệu đó.
4) Một số bài tập nâng cao khác (nếu có)

Bài 1: Viết chương trình con tính cos(x) theo công thức:

$$\cos(x) = \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + ...$$

Với độ chính xác ε= 0.0001

Sử dụng chương trình con để tính giai thừa

Bài 2: Viết chương trình con giải phương trình bậc 2: $Ax^2 + Bx + C = 0$, áp dụng tính với một ví dụ cho A, B, C bất kỳ nhập từ bàn phím. Nếu phương trình vô nghiệm, gán X1=X2=-999.

Thang chấm điểm bài thực hành

1. **Khai báo đầy đủ và chính xác** tất cả các biến sử dụng trong chương trình: 2đ, mỗi biến thiếu hoặc khai báo không đúng trừ 0,25đ. Quá 6 biến vi phạm sẽ không còn điểm phần này.
2. **Mở file, đọc hặc ghi dữ liệu vào file** đúng yêu cầu: 1đ, sai không có điểm.
3. **Trình bày các phần của chương trình** theo thứ tự đúng: 1đ. Bất cứ phần nào sắp xếp không đúng trừ 0,5đ.
4. **Nội dung**: chấm theo kết quả làm được, kết quả đúng trọn số điểm; ý tưởng đúng: ¼ số điểm, sử dụng lệnh đúng: ¼ nửa số điểm, được gợi ý và sửa đúng: ¼ số điểm.

- https://sites.google.com/site/nthanhbanvl/tin-hoc-lop-11-1/thuc-hanh

UNIVERSITY OF SCIENCE,

VIET NAM NATIONAL UNIVERSITY HO CHI MINH CITY

FINAL TEST

Module name:	**Application Programming**	**Code:**	OMH10003
Time:	**60 minute**	Date:	

Important Reminders: ***Students are allowed to use materials when doing test.***

Use the Fortran programming language to calculate the following:

Give the water level data in file C.TXT

a. (2.0 marks) Read a data file with 106 elements

b. (2.0 marks) In the original data series, there are unknown values numbered -999, completely eliminate these unknown values; Write the results in file KQ1.TXT.

c. (2.0 marks) Using the results in question b, calculate the height of wave H.

d. (2.0 marks) Calculate the significant wave height (average 1/3 of the largest wave height), write the results in file KQ2.TXT.

e. (2.0 marks) Write a subprogram to calculate the significant wave height.

Attention:

1. Fully and accurately declare the data types of all variables used in the chapter

For each missing variable or incorrectly declared data type, deduct 0.25 marks.

2. Open the file, read or write data to the file according to the requirements, otherwise there will be no points.

3. Present the parts of the program in the correct order, regardless of the arrangement incorrect minus 0.25 marks.

4. Content: Score according to the results achieved, correct results: full score; have ideas correct: 1/3 of the score, using the correct command: 1/3 of the score.

Answer

```
Program final_test
integer*4 n
parameter (n=106)
integer*4 i, j , m, k
real*4 a(n), H(n), tam, H_s
Open (unit=1, file='C.txt', status='old')
Open (unit=2, file='KQ1.txt', status='unknown')
Open (unit=3, file='KQ2.txt', status='unknown')
! -------------------------Question a -------------------------
do i=1, n
    read (1,*) a(i)
```

```
enddo
! -------------------------Question b -------------------------
m = 0
do i = 1, n
    if (a(i) = = -999) then
        a(i) = a(i+1)
        m = m+1
    endif
enddo
do i = 1, n – m
    write (2,*) a(i)
enddo
! -------------------------Question c -------------------------
k = 1
do i = 2, m, 2
    H(k) = abs (a(i)) + abs (a(i-1))
    k=k+1
enddo
! -------------------------Question d -------------------------
do i=1, k-2
    do j=i+1, k-1
        if (H(i) < H(j)) then
                tam = H(i)
                H(i) = H(j)
                H(j) = tam
        endif
    enddo
enddo
k = mod (k, 3)
do i=1, k
    H_s = H_s + H(i)
Enddo
H_s = H_s ./ k
Write (2, *) H_s
```

```
close (1)
close (2)
close (3)
! -------------------------Question e -------------------------
Contains
Subroutine Find_Hs (q, n1, HS)
integer*4 i1, j1, n1
real*4 q(n1), tam1, Hs
do i1=1, n1-1
    do j1=i1+1, n1
        if (q(i1) < q(j1)) then
                tam1 = q(i1)
                q(i1) = q(j1)
                q(j1) = tam1
        endif
    enddo
enddo
n1 = mod (n1, 3)
do i1=1, n1
    Hs = Hs + q(i1)
Enddo
Hs = Hs ./ n1
End
```

FINAL TEST

Student submissions

Nguyễn Văn Quang

```
Program Nguyen_Van_Quang
integer*4 i, j , m, k
real*4 a(106), H(106), tam, H_s
Open (unit=1, file='C.txt', status=' unknown ')
Open (unit=2, file='KQ1.txt', status='unknown')
Open (unit=3, file='KQ2.txt', status='unknown')
do i=1, 106
    read (1,*) a(i)
enddo
do i=1, 106
    read (*,*) a(i)
enddo

m = 0
do i = 1, 106
    if (a(i) = = -999) then
        a(i) = a(i+1)
        m = m+1
    endif
enddo
do i = 1, 106 – m
    write (2,*) a(i)
enddo

k = 1
do i = 2, m, 2
    H(k) = abs (a(i)) + abs (a(i-1))
    k=k+1
enddo

do i=1, k-2
```

```
    do j=i+1, k-1
        if (H(i) < H(j)) then
                tam = H(i)
                H(i) = H(j)
                H(j) = tam
        endif
    enddo
enddo
k = mod (k, 3)
do i=1, k
    H_s = H_s + H(i)
Enddo
H_s = H_s ./ k
Write (2, *) H_s

Contains
Subroutine Tim_Hs (D, E, H_1_3)
integer*4 i1, j1, E
real*4 D (n1), tam, H_1_3
do i1=1, E -1
    do j1=i1+1, E
        if (D (i1) < D (j1)) then
                tam= D (i1)
                D (i1) = D (j1)
                D (j1) = tam
        endif
    enddo
enddo
E = mod (E, 3)
do i1=1, E
    H_1_3 = H_1_3 + D(i1)
Enddo
H_1_3 = H_1_3 ./ E
End
```

FINAL TEST

Commnents

Good job, complete variable declaration and correct data type. However, students still have some minor errors: not closing previously opened files; Variables in subprogram and main program are the same

Marks: 9.0

UNIVERSITY OF SCIENCE,

VIET NAM NATIONAL UNIVERSITY HO CHI MINH CITY

FINAL TEST

Module name:	**Application Programming**	**Code:**	OMH10003
Time:	**60 minute**	Date:	

Important Reminders: ***Students are allowed to use materials when doing test.***

Question 1: Which of the following statements about *global* and *local variables* in Fortran is false?

A. The variable declared in the main program is a *global variable* and the variable declared in the subroutine is a *local variable*.

B. *Global and local variables* are both declared in the main program.

C. The main program will not be able to refer to *local variables* declared in the inner subroutines.

D. Internal subroutines are allowed to refer to *global variables* when *local variables* are not declared.

Question 2: Running the fortran program below, what is the result (related to accuracy)?

```
Program test
Real X
X = 23456789.0
Write (*,10) X
10 format (f30.2)
End
```

A. 23456789.00

B. 23456800.00

C. 23456790.00

D. 23456780.00

Question 3: If we run the program below, what output will we get?

```
Program test
Real c
c=2./3
Write(*,*)c
End
```

A. 6.666667E-1

B. 6.666666E-1

C. 0.000000E+00

D. 1.000000E+00

Question 4: In Fortran version 77, comments are written after the letter C, so in Fortran version 90, comments are written after what letter or mark?

A. Z

B. #

C. !

D. %

Question 5: In Fortran, what is the basic difference between an inner subroutine and an outer subroutine?

A. Inner subroutines can use variable names, constants, and declarations of the program unit that manages it, external subroutines do not.
B. External subroutines can use the variable names, constants, and declarations of the program unit that manages it, internal subroutines cannot.
C. The inner subroutine is a procedure (SUBROUTINE), the outer subroutine is a function (FUNCTION).
D. External subroutines are compiled together with the program unit on which it depends, while internal subroutines can be compiled independently.

Question 6: To solve the problem using the Fortran programming language, we need to perform the following 4 steps:
1) Edit the program's source code.
2) Analyze the problem, determine the algorithm, the steps and the sequence of steps.
3) Run the program (ie run the executable file) to get the result.
4) Compile the program.
The above 4 steps are not in order, please indicate the correct order of those 4 steps:
A. The order is: 2, 1, 4, 3
B. The order is: 2, 1, 3, 4
C. The order is: 1, 2, 4, 3
D. The order is: 1, 2, 3, 4

Question 7: There are 4 variable and constant names set as follows: 2A; An An; delta_4x; theta+alpha, what name is invalid in Fortran?
A. All 4 are not valid
B. 2A; An An; theta+alpha
C. 2A; delta_4x; theta+alpha
D. No invalid variable names

Question 8: In Fortran, each array is defined by what factors?
A. Array name, number of dimensions, array type and data type in array.
B. Array name, number of dimensions, maximum size and type of array.
C. Array name, number of dimensions, maximum size and arrangement of array elements.
D. Array name, number of dimensions, maximum size and data type in the array.

Question 9: Given 3 **DO** loops in Fortran as below, which one is the wrong structure?

(1)	(2)	(3)
Program test **Real** c C=0. **do** 10 i=1,10 C=C+i 10 **write**(*,*) c **End**	**Program** test **Real** c C=0. **do** i=1,10 C=C+i **enddo** **write**(*,*) c **End**	**Program** test **Real** c C=0. **do** 10 i=1,10 C=C+i 10 **continue** **write**(*,*) c **End**

A. Loop 1
B. Loop 3

C. Both loop 1 and 3
D. No loop is wrong

Question 10: Given the code as below:

```
Program test
Write(*,*) 'KQ1 =', INT(19/4),  ' ; ' , 'KQ2 =', INT(19./4)
End
```

Can you tell me the output on the screen?
A. KQ1 = 3; KQ2 = 3
B. KQ1 = 4; KQ2 = 4.75
C. KQ1 = 4; KQ2 = 5
D. KQ1 = 4; KQ2 = 4

Question 11: The first standard version of the Fortran language was released
A. First Fortran Version 1957
B. Fortran 66 (issued in 1966)
C. Fortran 77 (published in 1978)
D. Fortran 90 (issued in 1991)

Question 12: Let's run the fortran program below, what is the result?

```
Program test
real c
c=2/3
Write(*,*) c
 End
```

A. 6.666667E-1
B. 6.666666E-1
C. 0.000000E+00
D. 1.000000 E+00

Question 13: Given the array declaration below, what is the size of array B?

```
Integer B(-3:5, 2:13, 0:5)
```

A. 9 x 12 x 5
B. 8 x 12 x 6
C. 9 x 11 x 5
D. 9 x 12 x 6

Question 14: Run the fortran program below, what will the output look like?

```
Program test
Real a
Integer b, c
a=2.8
b=5
c=a+b
Write(*,*)'KQ1=', a+b,  ';' ,'KQ2=', c
End
```

A. KQ1 = 7; KQ2 = 7

B. KQ1 = 7.800000; KQ2 = 7.800000
C. KQ1 = 7.800000; KQ2 = 7
D. KQ1 = 7; KQ2 = 7.800000

Question 15: With the data type declared as below, how much memory is allocated for array A?

Real *A(10, 3, 4)*

A. 240 byte
B. 960 byte
C. 120 byte
D. 480 byte

Question 16: In Fortran, do the calculation below, what is the output?

```
Program test
Real A
A=10 / 3 * 3
Write(*,*) A
End
```

A. 1.111111
B. 10.000000
C. 9.000000
D. 1.000000

Question 17: Given the program below, what is the result of Y?

```
Program test
Real X, Y
X = 4.0
Y = 5.6 + SQRT(X)
Write(*,*) Y
End
```

A. 7.600000
B. 7.000000
C. 21.000000
D. 21.600000

Question 18: Determine which of the following constants is spelled incorrectly according to Fortran's convention?

a=9,87
b= .0
c= 25.82
d=-356231
e= 3.57E2.1
f= 3.57E+2
g=3,57E-2

A. a, e, g
B. b, d, g
C. a, b, e

D. b, d, e

Question 19: In Fortran, what is the output of the program below?

```
Program test
Real Y
Y = 10+3*2**4 -16/3
Write(*,*) Y
End
```

A. 1300.666667
B. 52.666667
C. 1301.000000
D. 53.000000

Question 20: Given the code below, the writer forgot to declare the variable, what is the output of C and D (see implicit type rule)?

```
Program test
Logan=0.12
Rick  =0.12
C=Logan*5 + 1.3
D=Rick*5+1.3
Write(*,*)'C=', C,  ';' ,'D=', D
End
```

A. C=1.300000; D=1.300000
B. C=1.900000; D=1.900000
C. C=1.300000; D=1.900000
D. C=1.900000; D=1.300000

Question 21: Let's run the Fortran program below, what is the maximum value of N input so that the program still gives the correct result?

```
Program test
Integer i, N, GT
Write(*,*) 'nhap gia tri cua N'
Read(*,*) N
GT=1
Do i=1,N
        GT=GT*i
Enddo
Write(*,*) GT
End
```

A. 15
B. 16
C. 17
D. 18

Question 22: Given the Fortran program below, which command line has a syntax error?

```
Program test                ! 1
Read(*,*) a                 ! 2
```

```
If (a==0) then              ! 3
      Write(*,*) 'a =0'     ! 4
Elseif (a<0) then           ! 5
      Write(*,*) 'a <0'     ! 6
Else (a>0) then             ! 7
      Write(*,*) 'a >0'     ! 8
Endif                       ! 9
End                         ! 10
```

A. 3rd Command Line

B. 5th command line

C. 7th Command Line

D. 9th command line

Question 23: In Fortran 90, if the command is too long to write all on one line, the user can move part of the statement to the next line. Then, what symbol or command must be used to connect the 2 lines together?

A. AND
B. &
C. CON
D. OR

Question 24: In Fortran 90, given the program below, what is the output to the screen?

```
Program test
Complex a
a=(-3,-4)
Write(*,*) abs(a)
End
```

A. 5.000000
B. 3.000000
C. 25.000000
D. 7.000000

Question 25: In Fortran, the **.LE.** operation. what do they mean?

A. Smaller
B. Greater than or equal to
C. Less than or equal to
D. Bigger

Question 26: In Surfer, the essence of taking steps in turn **Grid | Data** is the data to create a grid file (Creating a Grid File)?

A. Open an existing data file
B. Interpolate data onto each grid cell
C. Generate contour map
D. Map Digitization

Question 27: In Fortran 90, when will the below program stop?

```
Program test27
real C
read(*,*) C
Do while (C .NE. 0)
        Write(*,*) C
        Read (*,*) C
Enddo
End
```

A. C = 0
B. C ≠ 0
C. C > 0
D. C < 0

Question 28: Please indicate the output of the program below:

```
Program test27
Integer n
real C
n=1
C=n**2+10*n-7
Do while (C<200 )
        n=n+1
        C=n**2+10*n-7
Enddo
n=n-1
End
```

A. n =9
B. n =10
C. n =11
D. n = 12

Question 29: In the Fortran programming language, given the command line **I=1,12,3**, what is the value of I respectively?

A. 1, 4, 7, 10

B. 3, 6, 9, 12
C. 0, 3, 6, 9, 12
D. 1, 4, 7, 10, 13

Question 30: In the Fortran programming language, given the code below, what is the output of the variable A to the screen?

```
Program test2
Integer*4 I, A
Do i=1,10,2
        A=3*i
```

```
Enddo
 Write (*,*) A
End
```

A. 30
B. 27
C. 6
D. 3

Applied Programming

Semester	Code subject	Class	Student ID	Last name	First Name	Midterm	Final exam			Exercise + diligence	Total
							Theoretical	Practice	Average		
2	OMH10003	22HDH1	19110361	Chu Khánh	Linh		Absence	Absence			**0.00**
2	OMH10003	22HDH1	21110355	Nguyễn Thế Minh	Nhật		Absence	Absence			**0.00**
2	OMH10003	22HDH1	22210001	Đinh Hoàng Quốc	An	8.00	5.67	7.00	**6.33**	0.25	**7.25**
2	OMH10003	22HDH1	22210002	Ngô Đức Hoàng	Anh	7.50	4.33	6.00	**5.17**	0.00	**6.10**
2	OMH10003	22HDH1	22210003	Tống Vy	Anh	9.00	4.33	7.50	**5.92**	0.00	**7.15**
2	OMH10003	22HDH1	22210004	Lê Thị Hồng	Đào	10.00	4.33	9.00	**6.67**	0.25	**8.25**
2	OMH10003	22HDH1	22210005	Võ Ngọc	Diệp		Absence	Absence			**0.00**
2	OMH10003	22HDH1	22210006	Huỳnh Hân	Đình	10.00	5.67	8.50	**7.08**	0.25	**8.50**
2	OMH10003	22HDH1	22210007	Phan Thanh Hồng	Linh	6.00	3.67	8.00	**5.83**	0.25	**6.15**
2	OMH10003	22HDH1	22210008	Trần Ngọc Phương	Linh	8.00	3.00	9.00	**6.00**	0.00	**6.80**
2	OMH10003	22HDH1	22210009	Nguyễn Thị	Ngân	10.00	6.33	10.00	**8.17**	0.25	**9.15**
2	OMH10003	22HDH1	22210010	Phạm Thị Yến	Nhi	6.00	5.33	9.00	**7.17**	0.25	**6.95**
2	OMH10003	22HDH1	22210011	Lê Nguyễn Hạnh	Như	10.00	5.33	7.00	**6.17**	0.25	**7.95**
2	OMH10003	22HDH1	22210012	Nguyễn Văn	Quang	10.00	6.33	9.50	**7.92**	0.25	**9.00**
2	OMH10003	22HDH1	22210014	Nguyễn Đặng Thúy	Quỳnh	5.50	5.67	9.00	**7.33**	0.25	**6.85**
2	OMH10003	22HDH1	22210016	Trần Lê Minh	Thắng		Absence	Absence			**0.00**
2	OMH10003	22HDH1	22210017	Võ Thị Anh	Thi	10.00	4.67	5.00	**4.83**	0.25	**7.15**
2	OMH10003	22HDH1	22210018	Tống Anh	Thư		Absence	Absence			**0.00**
2	OMH10003	22HDH1	22210020	Nguyễn Nhật	Triết	10.00	5.67	9.00	**7.33**	0.25	**8.65**
2	OMH10003	22HDH1	22210021	Nguyễn Ngọc Phương	Vy	6.00	6.33	7.00	**6.67**	0.25	**6.65**
2	OMH10003	22HDH1	22210022	Lê Kim	Xuân	10.00	5.33	7.50	**6.42**	0.25	**8.10**

UNIVERSITY OF SCIENCE,

VIET NAM NATIONAL UNIVERSITY HO CHI MINH CITY

MID-TERM TEST

Module name:	**Application Programming**	**Code:**	OMH10003
Time:	**45 minute**	Date:	

Important Reminders: ***Students are allowed to use materials when doing test.***

Use the Fortran programming language to calculate the following:

a. (3.0 marks) Calculate string value

A = 1 + 2 – 3 + 4 – 5 + …….. + 100

b. (3.0 marks) Enter the number N from the keyboard, ask the user to re-enter until N is a positive integer greater than 2.

c. (3.0 marks) With the value N in question b, Calculate string values

B = 2 + 4 + 6+ 8+ …… + N

d. (1.0 marks) Write the results of question a, c to the screen.

Attention:

1. Fully and accurately declare the data types of all variables used in the chapter

For each missing variable or incorrectly declared data type, deduct 0.25 marks.

2. Open the file, read or write data to the file according to the requirements, otherwise there will be no points.

3. Present the parts of the program in the correct order, regardless of the arrangement incorrect minus 0.25 marks.

4. Content: Score according to the results achieved, correct results: full score; have ideas correct: 1/3 of the score, using the correct command: 1/3 of the score.

Answer

```
Program mid_test
Integer N, i
Real*4 A, B
A = 1
! -------------------------Question a -------------------------
Do i = 2, 100
    A = A + (i)*( (-1)**(i))
Enddo
! -------------------------Question b -------------------------
Read (*,*) N
do while (N < 2)
    Read (*,*) N
Enddo
```

```
! -------------------------Question c -------------------------
Do i = 2, N, 2
    B = B + i
Enddo
! -------------------------Question d -------------------------
Write (*,*) A, B
End
```

UNIVERSITY OF SCIENCE,

VIET NAM NATIONAL UNIVERSITY HO CHI MINH CITY

MIDDLE TEST

Student submissions

Nguyễn Văn Quang

```
Program Nguyen_Van_Quang
Integer N, i
Real*4 A, B
A = 1
Do i = 2, 100
    A = A + (i)*( (-1)**(i))
Enddo
Write (*,*) ‘ket qua cau A =’, A
Read (*,*) N
do while (N < 2)
    Read (*,*) N
Enddo
Do i = 2, N, 2
    B = B + i
Enddo
Write (*,*) ‘ket qua cau C, B =’, B
Write (*,*) A, B
End
```

Commnents

Students do well, understand the knowledge they have learned and apply it well to solve exercises

Marks: 10