Introduction to Object Oriented FORTRAN Programming Language (CSC 201)

Department of Computer Science

LECTURE 1

Basic concept of FORTRAN 2003 Programming

Basic concept of Programming

- Fortran has been the premier language for scientific computing since its introduction in1957.

- Fortran originally was designed to allow programmers to evaluate formulas. FORmula TRANslation—easily on large computers.

- Fortran compilers are now available on all sizes of machines, from small desktop computers to huge multiprocessors.

Basic concept of Programming

Different generations of Fortran:

1. FORTRAN66
2. FORTRAN77 (These two must not be used anymore!)
3. Fortran 90
4. Fortran 95
5. Fortran 2003 (**Object Oriented FORTRAN**)
6. Fortran 2008

Basic concept of Programming

The most important features introduced in Fortran 2003 are:

✓ interoperability with the C programming language , permitting easy portable access to the low-level facilities of C from Fortran programs and the portable use of Fortran libraries by programs written in C

✓ support for exceptions and IEEE arithmetic in so far as it does not conflict with existing Fortran arithmetic rules

✓support for object-oriented programming, including inheritance (type extension), polymorphism (dynamic typing), and type-bound procedures

✓data-type enhancements, such as parameterized derived types, allocatable components, and finalizers.

Basic concept of Programming

The most important features introduced in Fortran 2003 are:

✓input/output enhancements, such as user-defined derived-type input/output, asynchronous input/output, stream input/output, and the FLUSH statement to empty buffers

✓support for international usage, including the ISO 10646 character set and choice of a comma or period for the decimal symbol in numeric formatted input/output

✓ other features, such as procedure pointers, the PROTECTED and VOLATILE attributes, the IMPORT statement, access to environment variables and commandline arguments, better error handling, and better rounding control

Basic concept of Programming

Basic Concept of FORTRAN 2003 programming environment

✓A **Program** is an organized collection of program units. There must be exactly one main program, and in addition there may be modules, external subprograms, and block data units. Elements described by means other than Fortran may be included.

✓A **Module** provides a means of packaging related data and procedures, and hiding information not needed outside the module. There are several intrinsic modules.

✓The **Data Environment** consists of the data objects upon which operations will be performed to create desired results or values. These objects may have declared and dynamic types; they may have type parameters, and they may possess attributes such as dimensionality. They need not exist for the whole execution of the program. Allocatable objects and pointer targets may be created when needed and released when no longer needed.

Basic concept of Programming

Basic Concept of FORTRAN 2003 programming environment

✓**Program Execution** begins with the first executable construct in the main program and continues with successive constructs unless there is a change in the flow of control. When a procedure is invoked, its execution begins with its first executable construct. On normal return, execution continues where it left off. Execution may occur simultaneously with input/output processes.

✓The **Definition Status** of a variable indicates whether or not the variable has a value; the value may change during execution. Most variables are initially undefined and become defined when they acquire a value. The status also may become undefined during execution. Pointers have both an association status and a definition status. Allocatable objects have both an allocation status and a definition status.

✓**Scope** and **Association** determine where and by what names various entities are known and accessible in a program. These concepts form the information backbone of the language.

LECTURE 2

Program structure and layout

Program structure and layout

The program Statement

Each Fortran program begins with a program statement and ends with an end program statement. The **statement** consists of the keyword program followed by a **program name** of the programmer's choice. A name must start with a letter and consist of at most 63 letters, digits, and underscores; the letters may be uppercase or lowercase. Other names in Fortran also follow this rule.

The end program Statement

The **statement** begins with the keywords end program. It must be followed
by the name of the program. Every Fortran program must have an end program statement as its last statement

Program structure and layout

Structure of a FORTRAN Program

A Fortran program consists of one or more **program units**.

The forms of a program unit are:
main-program
module
external-subprogram
block-data

The form of a main program is:
[PROGRAM program-name]
[specification-part]
[execution-part]
[CONTAINS
internal-subprogram
[internal-subprogram] ...]
END [PROGRAM [program-name]]

The form of a module is:
MODULE module-name
[specification-part]
[CONTAINS
module-subprogram
[module-subprogram] ...]
END [MODULE [module-name]]

Program structure and layout

Structure of a FORTRAN Program

The form of a module subprogram and an external subprogram is:

subprogram-heading
[specification-part]
[execution-part]
[CONTAINS
internal-subprogram
[internal-subprogram] ...]
subprogram-ending

The form of an internal subprogram is:

subprogram-heading
[specification-part]
[execution-part]
subprogram-ending
]]

Program structure and layout

A Generic Structure of a FORTRAN Program

program name_of_the_program
! This is a comment
implicit none
type and variable definitions
.
.
.
executable statements
.
.
.
stop
contains *! Comment again*
local subroutine definitions
...
end program name_of_the_program

Program structure and layout

The first example is a program that prints the result of an addition.

```
program calculation_1
print *, 84 + 13
end program calculation_1
```

The program calculation_1 tells the computer to add the numbers 84 and 13 and
then to print the computed sum, 97. When the computer is told to run calculation_1,
it does precisely that: It adds the two numbers and prints their sum. T
Unlike many other programming languages, Fortran is *not*case-sensitive; that is,
program myprog
Program Myprog
PROGRAM myprog
and so forth, are all the same to the compiler. This is true for any name or statement in the program.

Program structure and layout

Printing both exact literal characters and a computed numeric value produces the
following easy-to-read output.

```
program calculation_1_v2
print *, "84 + 13 =", 84 + 13
end program calculation_1_v2
84 + 13 = 97
```

In the program calculation_1_v2 (calculation 1 version 2), there are two items in the list in the print statement, a character constant "84 + 13 =" to be printed exactly as written (but without the delimiting quotation marks) and an arithmetic expression whose value is first calculated and then printed.

Exercise: What computer output might be expected when the following program is run?

```
program simple
print *, 1, "and", 1, "equals", 1 + 1
end program simple
```