# CS 199 Computer Programming

Spring 2018

Lecture 7

One-dimension array

# Introduction

- Declare 1 variable to store a test score of 1 student.

  Integer :: score

- Declare 2 variables to store a test score of 2 students.

  Integer :: score1, score2

- Declare 100 variables to store a test score of 100 students.

  Integer :: score1, score2, … , score100

- Is it a smart ways? **NO**
- Solution: **Arrays**

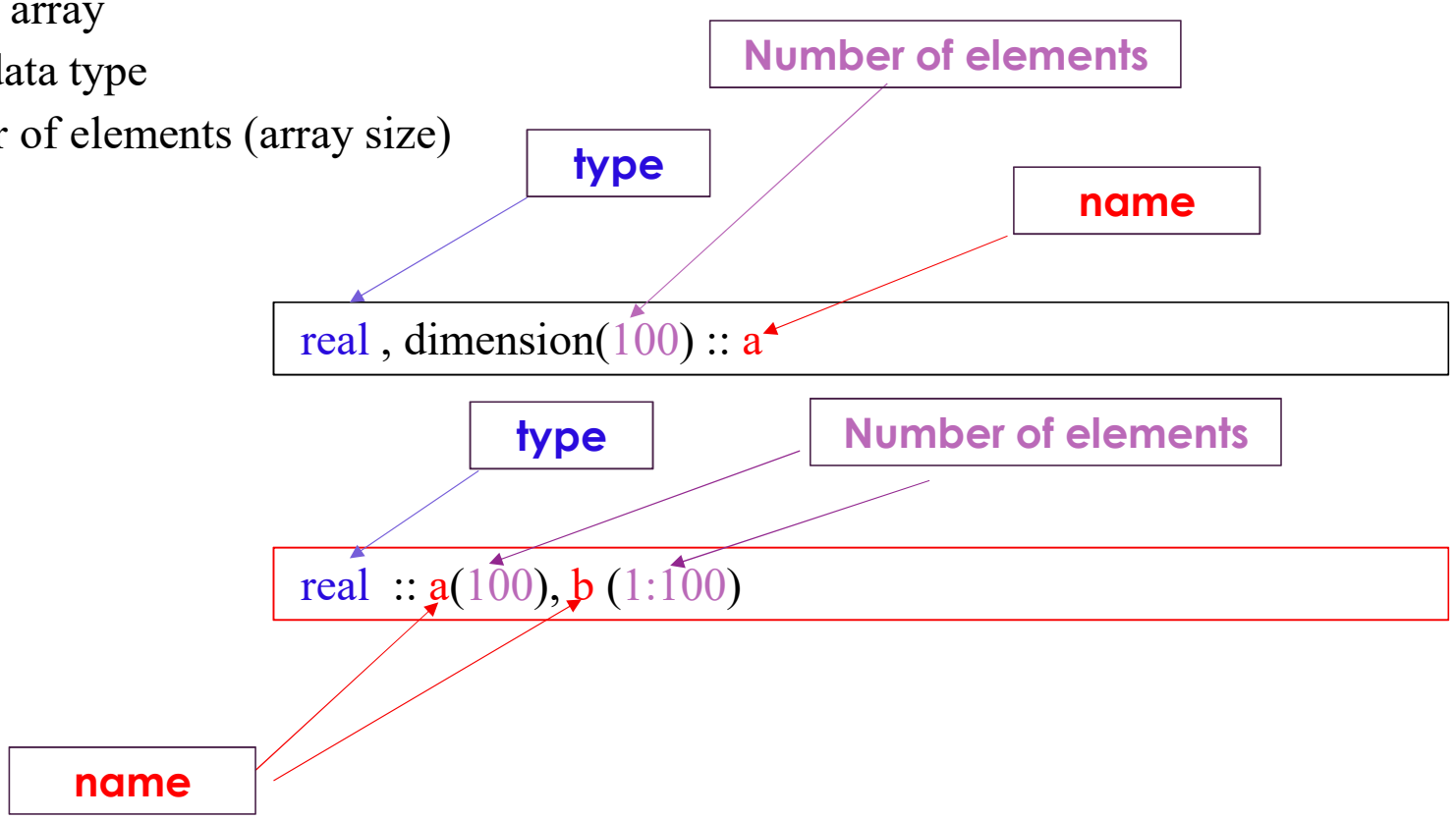
# Array

- An array a collection of a fixed number of components wherein all of the components have the same data type (e.g., a collection of integers, collection of characters, collection of real).

| Currency | U.S. $ | Aust $ | U.K. £ | Can $ | DMark | FFranc | ¥en | SFranc | Euro |
|---|---|---|---|---|---|---|---|---|---|
| Last Trade | N/A | Oct 14 | Oct 14 | Oct 14 | Oct 14 | Oct 14 | Oct 14 | Oct 14 | 12:39AM |
| U.S. $ | 1 | 0.6493 | 1.663 | 0.675 | 0.5513 | 0.1644 | 0.009316 | 0.6784 | 1.082 |
| Aust $ | 1.54 | 1 | 2.562 | 1.04 | 0.8491 | 0.2532 | 0.01435 | 1.045 | 1.666 |
| U.K. £ | 0.6012 | 0.3904 | 1 | 0.4058 | 0.3314 | 0.09883 | 0.005601 | 0.4079 | 0.6505 |
| Can $ | 1.481 | 0.9619 | 2.464 | 1 | 0.8167 | 0.2435 | 0.0138 | 1.005 | 1.603 |
| DMark | 1.814 | 1.178 | 3.017 | 1.224 | 1 | 0.2982 | 0.0169 | 1.231 | 1.963 |
| FFranc | 6.083 | 3.95 | 10.12 | 4.106 | 3.354 | 1 | 0.05667 | 4.127 | 6.582 |
| ¥en | 107.3 | 69.7 | 178.5 | 72.46 | 59.18 | 17.64 | 1 | 72.82 | 116.1 |
| SFranc | 1.474 | 0.9571 | 2.452 | 0.995 | 0.8126 | 0.2423 | 0.01373 | 1 | 1.595 |
| Euro | 0.9242 | 0.6001 | 1.537 | 0.6238 | 0.5095 | 0.1519 | 0.00861 | 0.627 | 1 |

# Declaring an Array

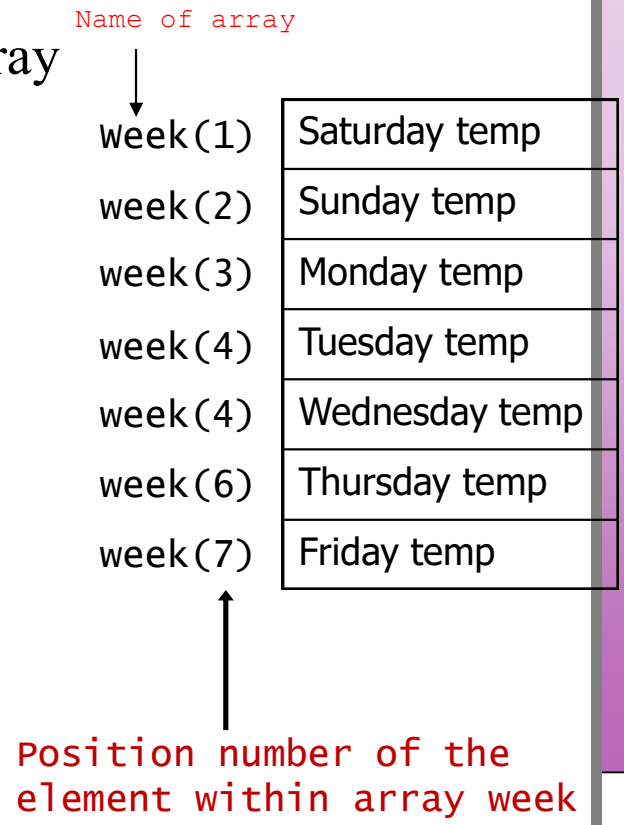
- When declaring arrays, programmer specify
  - Name
  - Type of array
    - Any data type
  - Number of elements (array size)
- Format



# Accessing an Element of the Array

- The variables making up the array are referred to as
  - **Indexed variables or Subscripted variables or Elements of the array**
- The number of indexed variables is equal to the size of the array
- To refer to an element, specify
  - Array name
  - Position number
- Format:

  **Arrayname(position number)**
  - First element at position 1
  - Last element at position n where n is the array size
  - 7 element array named week:
    - Week(1), week( 2 )...week(7)



# Array Element Manipulation

- Consider

**Integer :: i = 7, j = 2, k = 4, A(10)**

**A(1) = 1**

**A(i)= 5**

**A(j)= A(i)+ 3**

**A(j+1) = A(i)+ A(1)**

**A(A(j+1))= 12**

**Read (*,*) A(k)** **!where next input value is 3**

| 1 | 8 | 6 | 3 | - | 12 | 5 | - | - | - |
|---|---|---|---|---|---|---|---|---|---|
| A(1) | A(2) | A(3) | A(4) | A(5) | A(6) | A(7) | A(8) | A(9) | A(10) |

# Processing One-Dimensional Arrays

- Some basic operations performed on a one-dimensional array are:
  - Initializing
  - Inputting data
  - Outputting data stored in an array
  - Finding the largest and/or smallest element
- Each operation requires ability to step through the elements of the array
- Easily accomplished by a loop

# Processing One-Dimensional Arrays (cont.)

- Consider the declaration

```
integer ::  list(100)                    //array of size 100
```

- Initializing an array:

```
do i=1,100                               //Line 1
        list(i) =5                       //Line 2
```

- Reading data into an array

```
        read (*,*) list                  //Line 1
```

- Printing an array

```
        write (*,*) list                 //Line 1
```

Finding the sum and the average of the array

```
do i=1,100                               //Line 1
        sum=sum+list(i)                  //Line 2
Average = sum / 100                      //Line 3
```

# Some Restrictions on Array Processing

- Consider the following statements:

```
integer :: mylist(5) , yourlist(5)
```

- FORTRAN does not allow aggregate operations on an array:

```
yourlist = mylist;              //illegal
```

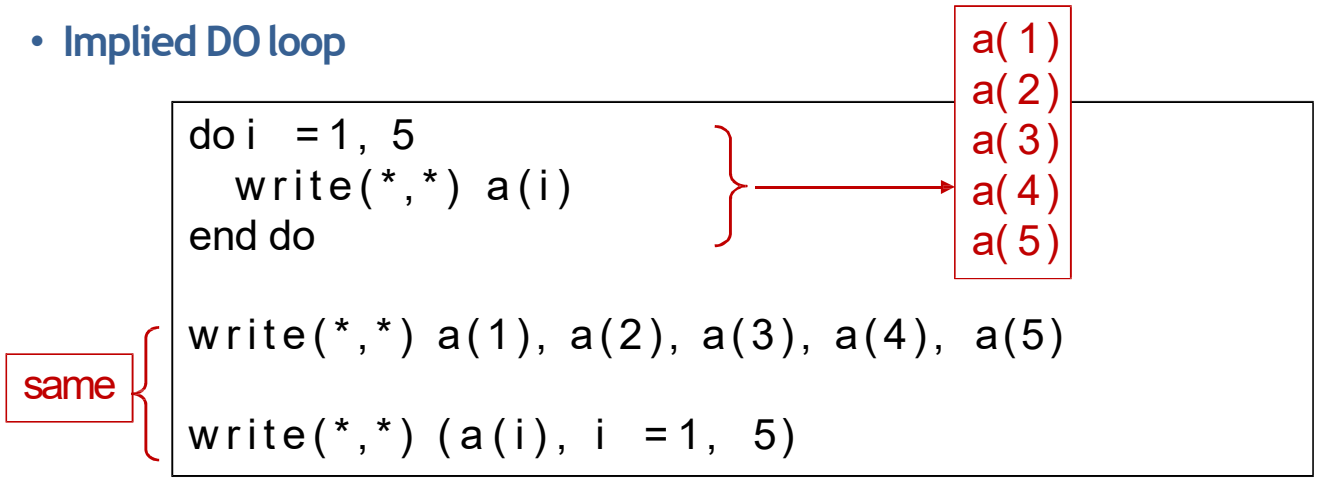
- Solution:

```
Do index = 1,5

    yourlist(index) = mylist(index)
```

# I/O of Arrays

- Implied DO loop



# arrays example

Euler noted that a sequence of 40 prime numbers `p` starting at 41 can be found from the formula:

$p = 41 + x + x^2$, for $0 \leq x \leq 39$

Write a program using an array constructor to store this sequence of 40 primes in an array

```
Integer  :: x(0:39),p(0:39)
do i=0,39
x(i)=i
P(i)=41+x(i)+x(i)*x(i)
end do
Write (*,*) p
end
```

# Array with Loops Example

```
!   finds minimum element of array
 integer:: numbers(5) ! array of numbers
integer :: minimum = 999999
write(*,*)  "Enter the array elements "
do i=1, 5
        read (*,*) numbers(i)
        if (numbers(i) < minimum)  then
                minimum=numbers(i)
        end if
End do
write (*,*) "The smallest number is: "   , minimum
Pause
end
```

```
Enter the numbers: 8 3 5 2 10
The smallest number is: 2
Press any key to continue...
```