# Function Overloading & Template

## Lab 12: Finding a number or a string

Week 13
Rung-Bin Lin

International Bachelor Program in Informatics
Yuan Ze University

## Purposes

- **Understand the followings**
  - ➢ **Function overloading**
  - ➢ **Function template**

## Function Overloading & Template

- Function overloading
  - C++ enables several functions of the same name to be defined, as long as they have different signatures.
  - Overloaded functions are normally used to perform similar operations that involve different program logic on different data types.
- If the program logic and operations are identical for each data type, overloading may be performed more compactly and conveniently by using function templates.

## Function Overloading

```
// Fig. 5.23: fig05_23.cpp
// Overloaded functions.
#include <iostream>
using namespace std;

// function square for int values
int square( int x )
{
   cout << "square of integer " << x << " is ";
   return x * x;
} // end function square with int argument

// function square for double values
double square( double y )
{
   cout << "square of double " << y << " is ";
   return y * y;
} // end function square with double argument

```

**Fig. 5.23** | Overloaded `square` functions. (Part 1 of 2.)

# Function Overloading cont.

```
int main()
{
   cout << square( 7 ); // calls int version
   cout << endl;
   cout << square( 7.5 ); // calls double version
   cout << endl;
} // end main
```

```
square of integer 7 is 49
square of double 7.5 is 56.25
```

**Fig. 5.23** | Overloaded `square` functions. (Part 2 of 2.)

# Function Template

```cpp
// Fig. 5.25: maximum.h
// Definition of function template maximum.
template < class T >  // or template< typename T >
T maximum( T value1, T value2, T value3 )
{
   T maximumValue = value1; // assume value1 is maximum

   // determine whether value2 is greater than maximumValue
   if ( value2 > maximumValue )
      maximumValue = value2;

   // determine whether value3 is greater than maximumValue
   if ( value3 > maximumValue )
      maximumValue = value3;

   return maximumValue;
} // end function template maximum
```

**Fig. 5.25** | Function template `maximum` header file.

# Function Template cont2.

```
// Fig. 5.26: fig05_26.cpp
// Function template maximum test program.
#include <iostream>
#include "maximum.h" // include definition of function template maximum
using namespace std;

int main()
{
   // demonstrate maximum with int values
   int int1, int2, int3;

   cout << "Input three integer values: ";
   cin >> int1 >> int2 >> int3;

   // invoke int version of maximum
   cout << "The maximum integer value is: "
      << maximum( int1, int2, int3 );

   // demonstrate maximum with double values
   double double1, double2, double3;

   cout << "\n\nInput three double values: ";
   cin >> double1 >> double2 >> double3;

```

**T** now is replaced by **int.**

Fig. 5.26 | Demonstrating function template maximum. (Part 1 of 2.)

# Function Template cont2.

```
    // invoke double version of maximum
    cout << "The maximum double value is: "
       << maximum( double1, double2, double3 );

    // demonstrate maximum with char values
    char char1, char2, char3;

    cout << "\n\nInput three characters: ";
    cin >> char1 >> char2 >> char3;

    // invoke char version of maximum
    cout << "The maximum character value is: "
       << maximum( char1, char2, char3 ) << endl;
} // end main
```

**T** now is replaced by **double.**

**T** now is replaced by **char.**

```
Input three integer values: 1 2 3
The maximum integer value is: 3

Input three double values: 3.3 2.2 1.1
The maximum double value is: 3.3

Input three characters: A C B
The maximum character value is: C
```

**Fig. 5.26** | Demonstrating function template `maximum`. (Part 2 of 2.)

## LAB 12 : Finding a number or a word

- Read a list of integers, real numbers, or strings which are being stored in the array **anAry[] in order of being read from the keyboard**. Find out the second largest number (string) and the index of its last occurrence in the array. Assume that a string consists of only English alphabets either in lower or upper case and the number of elements in the array is not larger than 10000. The strings are ordered in their lexicographic order. For example, A<a, a<b, aa<ab, aa<aaa, etc. check https://en.wikipedia.org/wiki/Lexicographic_order for details.

- Create the following function template for performing this task.

**void findSecondLargest(T anAry[], int numElm, T &secondLargest, int &loc);**

**The first parameter anAry[]    is an array that stores the data being processed. The second parameter is the number of elements actually stored in the array. If there is no second largest element, the fourth parameter loc is set to -1. Otherwise, loc is the index of the last second largest element in the array and the third parameter is the value of the second largest element.**

**Requirement: The content of anAry[] should not be changed.**

**NOTE: A template must be defined before it is used. So it has to be placed at a position before main() function. Also if we have a function template, we should not include its function prototype in the program. Please refer to the example in Fig. 5.25 and Fig. 5.26.**

## main() function

- An implementation of the main() function.

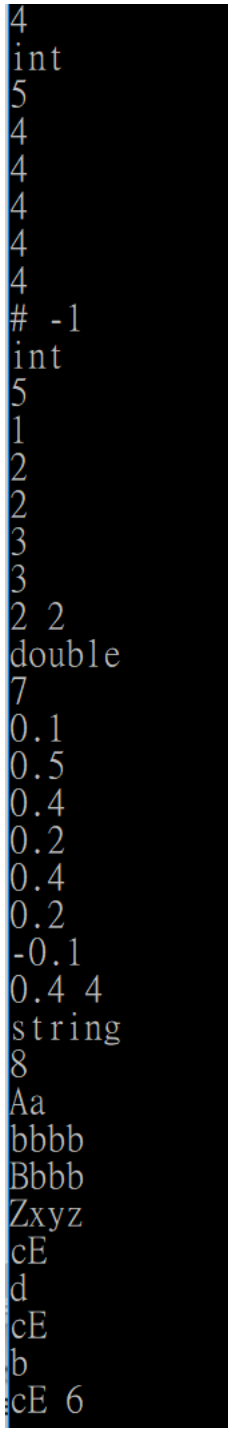
```
read in an Integer which specifies the number of test cases.
for each test case do {
      read in the data type, either int, double, or string.
      if data type is int do {
          read in an integer which specifies the number of elements in a test case.
          read the elements into an array (the array should have the right type).
          call findSecondLargest(…); // should provide proper arguments
          print out the result;
      }
      else if data type is double do {
          do things similar to that for data type being int.
      }
      else if data type is string do {
          do things similar to that for data type being int.
      }
      else return 1;
 } // end of for statement
 return 0;
```

# Input & Output

- **Input format**
  - **The first line specifies the number of test cases. Starting from the second line, input data for each test case are presented.**
  - **The first line of each test case specifies the data type of elements being read. It is *int* for integer data type, *double* for double precision data type, and *string* for string data type. The second line specifies the number of elements given for the test case. After this, each line gives an element of data being stored in the array.**
- **Output format**
  - **Each test case has one line which consists of two items. The first item is the value of the second largest element found in a test case. It is a # if there is no second largest element. The second item is the index of its last occurrence.**

# Example of Input & Output

| **Input** | **Output** |
|---|---|
| 4<br>int<br>5<br>4<br>4<br>4<br>4<br>4<br>int<br>5<br>1<br>2<br>2<br>3<br>3<br>double<br>7<br>0.1<br>0.5<br>0.4<br>0.2<br>0.4<br>0.2<br>-0.1<br>string<br>8<br>Aa<br>bbbb<br>Bbbb<br>Zxyz<br>cE<br>d<br>cE<br>b | # -1<br>2 2<br>0.4 4<br>cE 6 |



```
4
int
5
4
4
4
4
4
# -1
int
5
1
2
2
3
3
2 2
double
7
0.1
0.5
0.4
0.2
0.4
0.2
-0.1
0.4 4
string
8
Aa
bbbb
Bbbb
Zxyz
cE
d
cE
b
cE 6
```

## For TA Grading

- The main() function should be able to process the given input data all correctly.
- The main() function should make three calls to the function template *findSecondLargest(…)*.
- Check whether the function template *findSecondLargest(…)* is written. No other function is written.
- Should ask how to find the second largest element and the index of its last occurrence.
- Certainly, output should be correct.