Functions with Pass-by-Reference

Lab 10: Guessing a Password

Week 10

Rung-Bin Lin

International Bachelor Program in Informatics
Yuan Ze University

Purposes of the Lab

- Understanding the followings:
  - ➢ Pass by value & pass by reference
  - ➢ Reference type

Reference Type

```cpp
// Fig. 5.19: fig05_19.cpp
// Initializing and using a reference.
#include <iostream>
using namespace std;

int main()
{
   int x = 3;
   int &y = x; // y refers to (is an alias for) x

   cout << "x = " << x << endl << "y = " << y << endl;
   y = 7; // actually modifies x
   cout << "x = " << x << endl << "y = " << y << endl;
} // end main
```

```
x = 3
y = 3
x = 7
y = 7
```

Fig. 5.19 | Initializing and using a reference.

```cpp
// Fig. 5.20: fig05_20.cpp
// References must be initialized.
#include <iostream>
using namespace std;

int main()
{
   int x = 3;
   int &y; // Error: y must be initialized

   cout << "x = " << x << endl << "y = " << y << endl;
   y = 7;
   cout << "x = " << x << endl << "y = " << y << endl;
} // end main
```

Fig. 5.20 | Uninitialized reference causes a compilation error. (Part 1 of 2.)

Pass-by-Value vs. Reference

```cpp
// Fig. 5.18: fig05_18.cpp
// Comparing pass-by-value and pass-by-reference with references.
#include <iostream>
using namespace std;

int squareByValue( int ); // function prototype (value pass)
void squareByReference( int & ); // function prototype (reference pass)

int main()
{
   int x = 2; // value to square using squareByValue
   int z = 4; // value to square using squareByReference

   // demonstrate squareByValue
   cout << "x = " << x << " before squareByValue\n";
   cout << "Value returned by squareByValue: "
      << squareByValue( x ) << endl;
   cout << "x = " << x << " after squareByValue\n" << endl;

   // demonstrate squareByReference
   cout << "z = " << z << " before squareByReference" << endl;
   squareByReference( z );
   cout << "z = " << z << " after squareByReference" << endl;
} // end main
```

Fig. 5.18 | Passing arguments by value and by reference. (Part 1 of 2.)

```
// squareByValue multiplies number by itself, stores the
// result in number and returns the new value of number
int squareByValue( int number )
{
   return number *= number; // caller's argument not modified
} // end function squareByValue

// squareByReference multiplies numberRef by itself and stores the result
// in the variable to which numberRef refers in function main
void squareByReference( int &numberRef )
{
   numberRef *= numberRef; // caller's argument modified
} // end function squareByReference
```

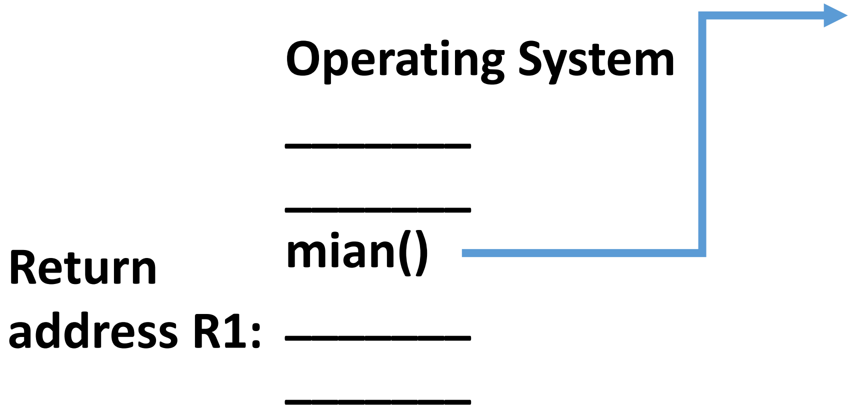
```
x = 2 before squareByValue
Value returned by squareByValue: 4
x = 2 after squareByValue

z = 4 before squareByReference
z = 16 after squareByReference
```

Fig. 5.18 | Passing arguments by value and by reference. (Part 2 of 2.)

Activation Records

- **When a function is called, an activation record (AR) is pushed into a *stack*. After executing the function, the activation record is popped (removed) from the stack. Stack is a piece of Last-in-first-out memory. Data can only be stored or retrieved from the stack.**



```
Int main(){
Int x=10;
Int number = 20;
squareByValue(x);  // call_1
squareByValue(number); // call_2
squareByReference(x); // call_3
squareByReference(number); // call_4
Return 0;
}
Int squareByValue (int number){
return number*number;
}

void squareByReference(int &numberRef){
numberRef = numberRef * numberRef;
}
```

AR of main()

Return address R1	
x	10
number	20

More on Pass-by-Value vs. Reference (1)

```
              Int main(){
              Int x = 10 , number = 20;
              squareByValue(x);  // call_1
Ret addr R2:  squareByValue(number); // call_2
Ret addr R3:  squareByReference(x); // call_3
Ret addr R4:  squareByReference(number); // call_4
Ret addr R5:  return 0; }

              Int squareByValue (int number){
              return number*number; }

              void squareByReference(int &numberRef){
              numberRef = numberRef * numberRef; }
```

Making call_1

AR of call_1	Return address R2
	number 10
AR of main()	Return address R1
	x 10
	number 20

After executing call_1

AR of main()	Return address R1
	x 10
	number 20

Making call_2

AR of call_2	Return address R3
	number 20
AR of main()	Return address R1
	x 10
	number 20

After executing call_2

AR of main()	Return address R1
	x 10
	number 20

Also see https://courses.washington.edu/css342/zander/css332/passby.html for another example.

More on Pass-by-Value vs. Reference (2)

```
               Int main(){
               Int x = 10 , number = 20;
               squareByValue(x);  // call_1
Ret addr R2:   squareByValue(number); // call_2
Ret addr R3:   squareByReference(x); // call_3
Ret addr R4:   squareByReference(number); // call_4
Ret addr R5:   return 0; }

               Int squareByValue (int number){
               return number*number; }

               void squareByReference(int &numberRef){
               numberRef = numberRef * numberRef; }
```

Making call_3

AR of call_3	Return address R4
	Address of *x* in main() for *numberRef*
AR of main()	Return address R1
	x 10
	number 20

After executing call_4

AR of main()	Return address R1
	x 100
	number 400

Making call_4

AR of call_4	Return address R5
	Address of *number* in main() for *numberRef*
AR of main()	Return address R1
	x 10
	number 20

After executing call_3

AR of main()	Return address R1
	x 100
	number 20

Lab 10: Guess a Password

- Write a program that will guess a password as follows:
  - **You are given a function void generatePassWd(string& passWd, int&) to generate a password that contains at most four lower-case letters, for example "abcd". The first parameter in this function contains a password. The second parameter is the length of the password.**
  - **Read from a keyboard a string which is the guess you made. Print out "Too high" if the string read from the keyboard is greater than passWd or print out "Too low" if it is smaller than passWd. Strings are compared in terms of their lexicographic order. For example, a < b, aa < ab, abc < abca, ect.**
  - **You should continue to read strings from the keyboard until you guess the password right. That is, the password read from the keyboard is the same as passWd.**
  - **If a right guess is made, print "Bravo, you guess it right!". Moreover, if the number of guesses you made for a right guess is smaller than or eqial to $\lceil \log_2 26^{len} \rceil$, then print out "You know the secret!", where len is the length of a password and $\lceil \quad \rceil$ is the *ceiling* function (page 194). Otherwise, print out "You should be able to do better." Here, you should use log2() function rather than log() or log10().**

void generatePassWd(string&, int&);

```
void generatePassWd(string &passWd, int &passLen){
    srand(time(0));
    passLen = rand()%4+1;
    for (int i=0; i<passLen; i++)
        passWd[i] = 'a' + rand()% 26;
}
```

Requirements (1)

- Write a function

  void guess(status &, string, int);

  where status is an enumeration type:

  enum status {TH, TL, RT}; // TH: too high; TL: too low; RT: right

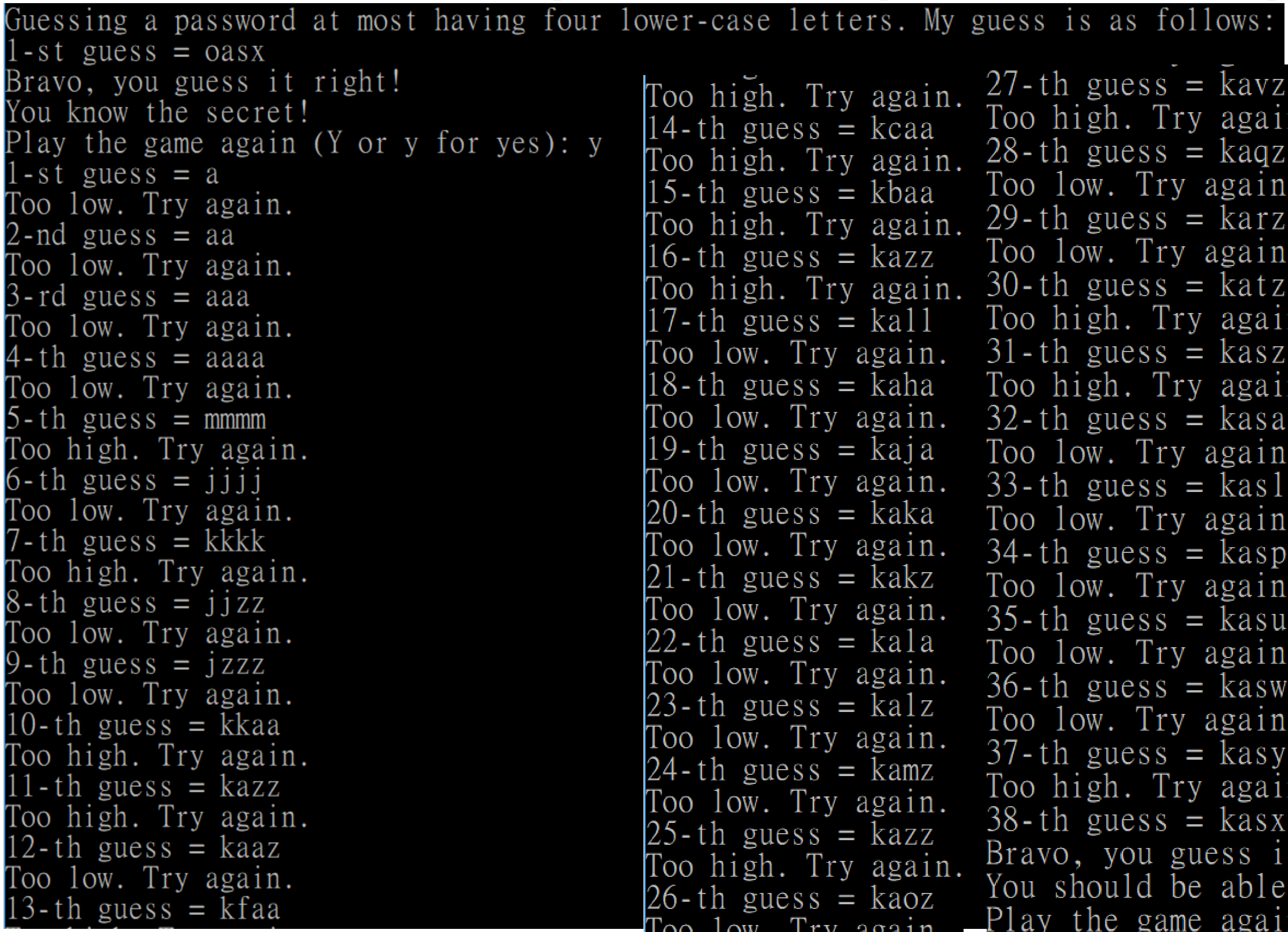
  The first parameter in guess(status &, string, int) is the guess result. The second parameter is the string we would like to guess. The third parameter is the length of a password. The return type should be void.
- This function should employ pass-by-reference to pass the guess result back to main() function. The main() function should have calls to this function as follows:

```
Int main()
{
  status aGuess;
  string passWd; // you may have to initialize it with a fixed length
.....
  guess(aGuess, passWd, passLen);
.....
}
```

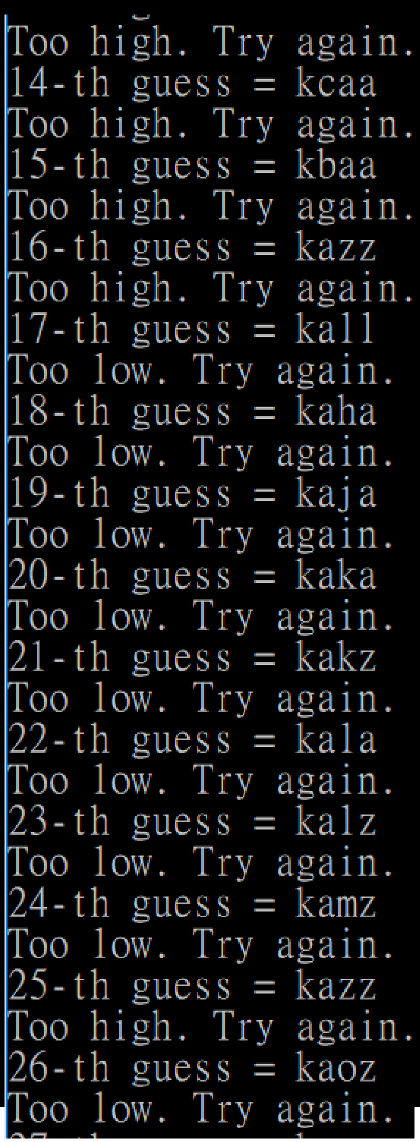
Requirements (2)

- **After a right guess is made, your program should ask whether to play the game again by presenting a prompt message “Play the game again (Y or y for yes): “. Otherwise, the program terminates.**

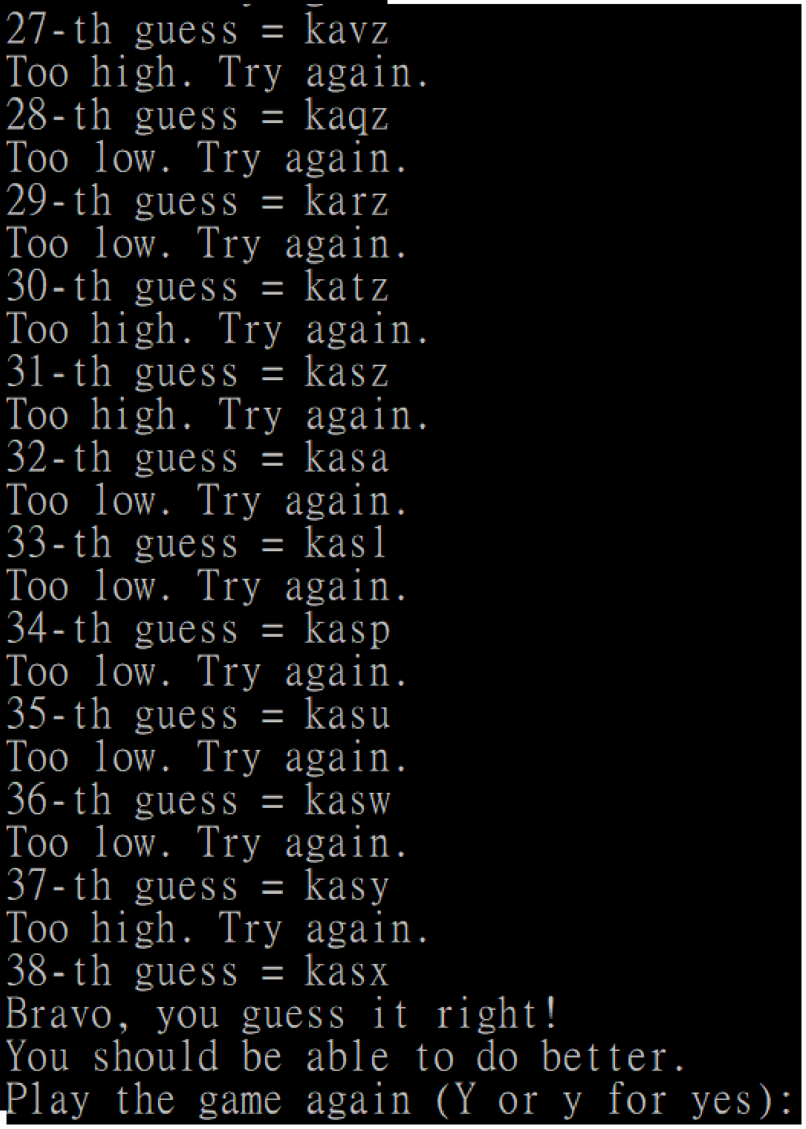
Example of Input & Output



```
Guessing a password at most having four lower-case letters. My guess is as follows:
1-st guess = oasx
Bravo, you guess it right!
You know the secret!
Play the game again (Y or y for yes): y
1-st guess = a
Too low. Try again.
2-nd guess = aa
Too low. Try again.
3-rd guess = aaa
Too low. Try again.
4-th guess = aaaa
Too low. Try again.
5-th guess = mmmm
Too high. Try again.
6-th guess = jjjj
Too low. Try again.
7-th guess = kkkk
Too high. Try again.
8-th guess = jjzz
Too low. Try again.
9-th guess = jzzz
Too low. Try again.
10-th guess = kkaa
Too high. Try again.
11-th guess = kazz
Too high. Try again.
12-th guess = kaaz
Too low. Try again.
13-th guess = kfaa
```



```
Too high. Try again.
14-th guess = kcaa
Too high. Try again.
15-th guess = kbaa
Too high. Try again.
16-th guess = kazz
Too high. Try again.
17-th guess = kall
Too low. Try again.
18-th guess = kaha
Too low. Try again.
19-th guess = kaja
Too low. Try again.
20-th guess = kaka
Too low. Try again.
21-th guess = kakz
Too low. Try again.
22-th guess = kala
Too low. Try again.
23-th guess = kalz
Too low. Try again.
24-th guess = kamz
Too low. Try again.
25-th guess = kazz
Too high. Try again.
26-th guess = kaoz
Too low. Try again.
```



```
27-th guess = kavz
Too high. Try again.
28-th guess = kaqz
Too low. Try again.
29-th guess = karz
Too low. Try again.
30-th guess = katz
Too high. Try again.
31-th guess = kasz
Too high. Try again.
32-th guess = kasa
Too low. Try again.
33-th guess = kasl
Too low. Try again.
34-th guess = kasp
Too low. Try again.
35-th guess = kasu
Too low. Try again.
36-th guess = kasw
Too low. Try again.
37-th guess = kasy
Too high. Try again.
38-th guess = kasx
Bravo, you guess it right!
You should be able to do better.
Play the game again (Y or y for yes):
```

Note: You should have 1-st, 2-nd, 3-rd, 4-th, …, 10-th, 11-th, 12-th, 13-th, …, 20-th, 21-th, ….