Programming Language (630203)
Fall 2010/2011 – Lecture Notes # 18

Value and reference parameters

Objectives of the Lecture

- **Value and reference parameters**
- **Reference parameters and value-returning functions**

- **Value parameter**: a formal parameter that receives a copy of the content of corresponding actual parameter
- **Reference parameter**: a formal parameter that receives the location (memory address) of the corresponding actual parameter

EXAMPLE 7-2

```
void expfun(int one, int& two, char three, double& four)
{
    .
    .
    .
}
```

The function `expfun` has four parameters: (1) `one`, a value parameter of type `int`; (2) `two`, a reference parameter of type `int`; (3) `three`, a value parameter of type `char`; and (4) `four`, a reference parameter of type `double`.

Value Parameters

- If a formal parameter is a value parameter
  - The value of the corresponding actual parameter is copied into it
- The value parameter has its own copy of the data
- During program execution
  - The value parameter manipulates the data stored in its own memory space

Reference Variables as Parameters

- If a formal parameter is a reference parameter
  - It receives the memory address of the corresponding actual parameter
- A reference parameter stores the address of the corresponding actual parameter
- During program execution to manipulate data
  - The address stored in the reference parameter directs it to the memory space of the corresponding actual parameter
- Reference parameters can:
  - Pass one or more values from a function
  - Change the value of the actual parameter

- Reference parameters are useful in three situations:
  - Returning more than one value
  - Changing the actual parameter
  - When passing the address would save memory space and time

Reference Parameters and Value-Returning Functions

- You can also use reference parameters in a value-returning function
  - Not recommended
- By definition, a value-returning function returns a single value
  - This value is returned via the return statement
- If a function needs to return more than one value, you should change it to a void function and use the appropriate reference parameters to return the values

```
//Example 1 illustrating how two variables are swapped.
#include <iostream>
using namespace std;
void Swap (int &x, int &y);
int main ()
{
   int x1,y1;
   x1 =10;
   y1 =15;
   cout << "before calling the Swap function"
        << x1 << "\t" << y1 << endl;
   Swap(x1,y1);
   cout << "after calling the Swap function"
        << x1 << "\t" << y1 << endl;

   return 0;
}
void Swap (int &x, int &y)
{
   int temp;
   cout << "in Swap function before changing"
        << x << "\t" << y << endl;
   temp = x;
   x = y;
   y = temp;
   cout << "in Swap function after changing"
        << x << "\t" << y << endl;
}
//Example 2 illustrating how a value parameter works.
#include <iostream>
using namespace std;
void funcValueParam(int num);
int main()
{
    int number = 6;                                    //Line 1
    cout << "Line 2: Before calling the function "
```

```
         << "funcValueParam, number = " << number
         << endl;                                        //Line 2
    funcValueParam(number);                              //Line 3
    cout << "Line 4: After calling the function "
         << "funcValueParam, number = " << number
         << endl;                                        //Line 4
   return 0;
}
void funcValueParam(int num)
{
    cout << "Line 5: In the function funcValueParam, "
         << "before changing, num = " << num
         << endl;                                        //Line 5
    num = 15;                                            //Line 6
    cout << "Line 7: In the function funcValueParam, "
         << "after changing, num = " << num
         << endl;                                        //Line 7
}
//Example 3 This program reads a course score and prints the
//associated course grade.
#include <iostream>
using namespace std;
void getScore(int& score);
void printGrade(int score);
int main ()
{
    int courseScore;
    cout << "Line 1: Based on the course score, \n"
         << "        this program computes the "
         << "course grade." << endl;                     //Line 1
    getScore(courseScore);                               //Line 2
    printGrade(courseScore);                             //Line 3
    return 0;
}
void getScore(int& score)
{
    cout << "Line 4: Enter course score: ";              //Line 4
    cin >> score;                                        //Line 5
    cout << endl << "Line 6: Course score is "
         << score << endl;                               //Line 6
}
void printGrade(int cScore)
{
    cout << "Line 7: Your grade for the course is ";     //Line 7
    if (cScore >= 90)                                    //Line 8
        cout << "A." << endl;
    else if (cScore >= 80)
        cout << "B." << endl;
    else if(cScore >= 70)
```

```
        cout << "C." << endl;
    else if (cScore >= 60)
        cout << "D." << endl;
    else
        cout << "F." << endl;
}
```

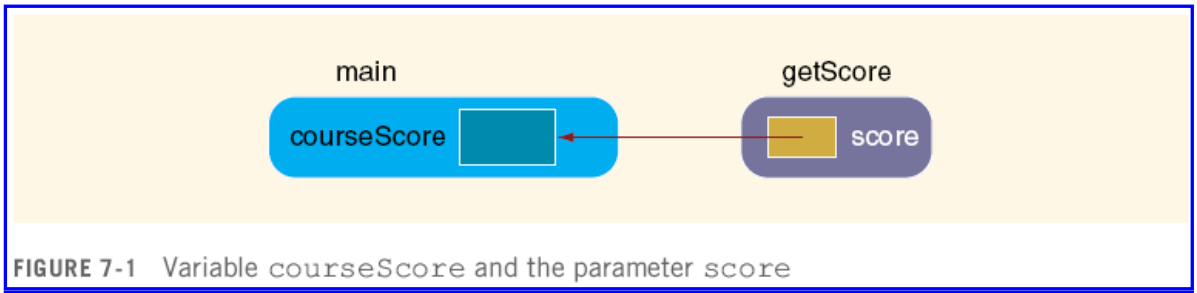

FIGURE 7-1 Variable courseScore and the parameter score

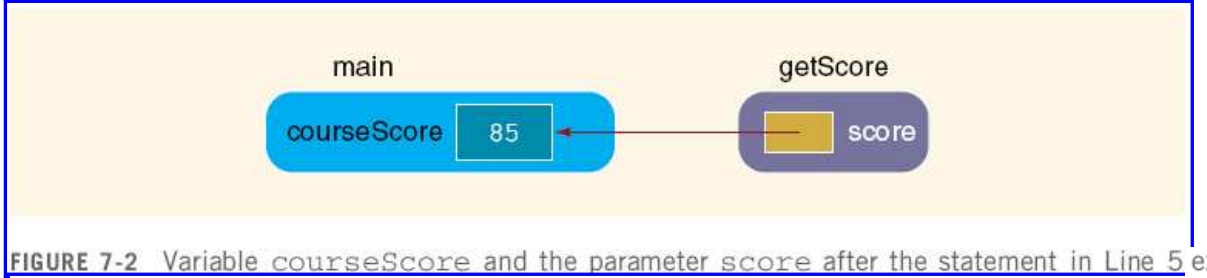

FIGURE 7-2 Variable courseScore and the parameter score after the statement in Line 5 executes

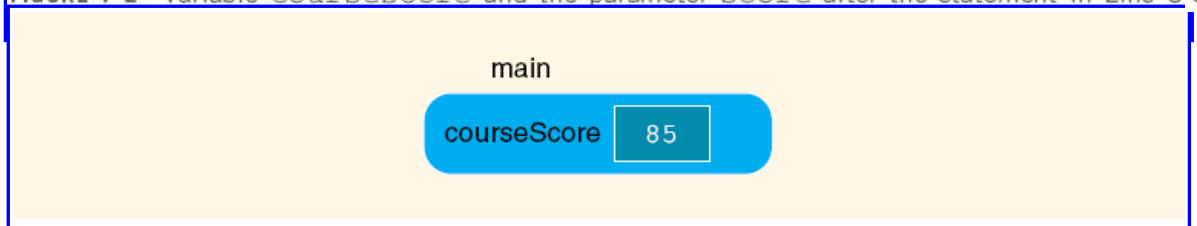

FIGURE 7-3 Variable courseScore after the statement in Line 6 is executed and control goes back to main

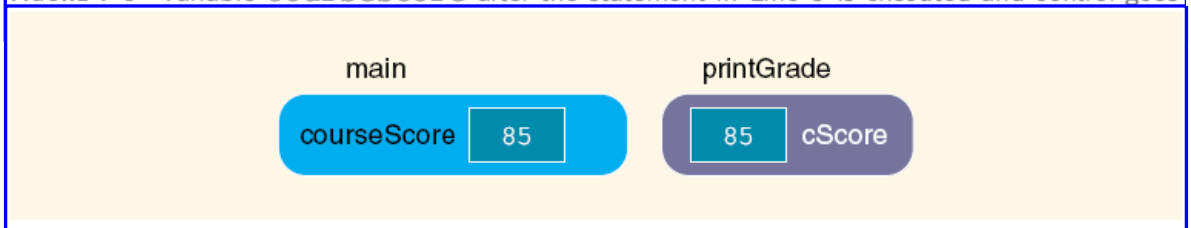

FIGURE 7-4 Variable courseScore and the parameter cScore