6.13 Function name overloading

Sometimes a program has two functions with the same name but differing in the number or types of parameters, known as ***function name overloading*** or just ***function overloading***. The following two functions print a date given the day, month, and year. The first function has parameters of type int, int, and int, while the second has parameters of type int, string, and int.

Figure 6.13.1: Overloaded function name.

```
#include <iostream>
#include <string>
using namespace std;

void DatePrint(int currDay, int currMonth, int currYear) {
   cout << currMonth << "/" << currDay << "/" << currYear;
}

void DatePrint(int currDay, string currMonth, int currYear) {
   cout << currMonth << " " << currDay << ", " << currYear;
}

int main() {

   DatePrint(30, 7, 2012);
   cout << endl;

   DatePrint(30, "July", 2012);
   cout << endl;

   return 0;
}
```

```
7/30/2012
July 30, 2012
```

Feedback?

The compiler determines which function to call based on the argument types. DatePrint(30, 7, 2012) has argument types int, int, int, so calls the first function. DatePrint(30, "July", 2012) has argument types int, string, int, so calls the second function.

More than two same-named functions is allowed as long as each has distinct parameter types. Thus, in the above program:

- DatePrint(int month, int day, int year, int style) can be added because the types int, int, int, int differ from int, int, int, and from int, string, int.
- DatePrint(int month, int day, int year) yields a compiler error, because two functions have types int, int, int (the parameter names are irrelevant).

A function's return type does not influence overloading. Thus, having two same-named function definitions with the same parameter types but different return types still yield a compiler error.

The use of overloading and of default parameter values may be combined as long as no ambiguity is introduced. Adding the function `void DatePrint(int month, int day, int year, int style = 0)` above would generate a compiler error because the compiler cannot determine if the function call `DatePrint(7, 30, 2012)` should go to the "int, int, int" function or to that new "int, int, int, int" function with a default value for the last parameter.

PARTICIPATION ACTIVITY 6.13.1: Function name overloading.

Given the following function definitions, type the number that each function call would print. If the function call would not compile, choose Error.

```
void DatePrint(int day, int month, int year) {
   cout << "1" << endl;
}

void DatePrint(int day, string month, int year) {
   cout << "2" << endl;
}

void DatePrint(int month, int day) {
   cout << "3" << endl;
}
```

1) DatePrint(30, 7, 2012);

- 1
- 2
- 3
- Error

2) DatePrint(30, "July", 2012);

- 1
- 2
- 3
- Error

3) DatePrint(7, 2012);

- 1
- 2
- 3
- Error

4) DatePrint(30, 7);

- 1
- 2
- 3
- Error

5) DatePrint("July", 2012);

- 1
- 2
- 3
- Error

Feedback?

Exploring further:

- Overloaded functions from cplusplus.com.

CHALLENGE ACTIVITY 6.13.1: Overload salutation printing.

Complete the second PrintSalutation function to print the following given personName "Holly" and customSalutation "Welcome":

```
Welcome, Holly
```

End with a newline.

```
#include <iostream>
#include <string>
using namespace std;

void PrintSalutation(string personName) {
   cout << "Hello, " << personName << endl;
}

// Define void  PrintSalutation(string personName, string customSalutation)...

/* Your solution goes here  */
   void  PrintSalutation(string personName, string customSalutation){
      cout << customSalutation << ", "<<personName<< endl;
   }

int main() {
```

```
   PrintSalutation("Holly", "Welcome");
   PrintSalutation("Sanjiv");

   return 0;
}
```

Run ✓ All tests passed

✓ Testing with two string arguments: Holly, Welcome

Your output `Welcome, Holly`

✓ Testing with one string argument: Sanjiv

Your output `Hello, Sanjiv`

Feedback?

CHALLENGE ACTIVITY 6.13.2: Convert a height into inches.

Write a second ConvertToInches() with two double parameters, numFeet and numInches, that returns the total number of inches. Ex: ConvertToInches(4.0, 6.0) returns 54.0 (from 4.0 * 12 + 6.0).

```
#include <iostream>
using namespace std;

double ConvertToInches(double numFeet) {
   return numFeet * 12.0;
}

/* Your solution goes here  */
double ConvertToInches(double numFeet, double numInches) {
   return numFeet * 12.0+numInches;
}

int main() {
   double totInches;

   totInches = ConvertToInches(4.0, 6.0);
   cout << "4.0, 6.0 yields " << totInches << endl;

   totInches = ConvertToInches(5.8);
   cout << "5.8 yields " << totInches << endl;
   return 0;
```

Run ✓ All tests passed

✓ Testing with two arguments: 4.0, 6.0

Your value `54`

✓ Testing with one argument 5.8

Your value `69.6`

Feedback?