# Activity #14: Variable Scope

## Recorder's Report

Manager: Reader:

Recorder: Driver:

Date: Score: Satisfactory / Not Satisfactory

Record your team's answers to the key questions (marked with  ) below.

a) Model 1, Question #7

b) Model 2, Question #13

c) Model 3, Question #17

# Activity #14: Variable Scope

In this activity, you will work in teams of 3–4 students to learn new concepts. This activity focuses on variable scope in C++.

## Content Learning Objectives

*After completing this activity, students should be able to:*

- Explain the difference between global and local variables in C++
- Identify the scope of a variable in a C++ code snippet
- Recognize variable shadowing and masking in C++ code

## Process Skill Goals

*During the activity, students should make progress toward:*

- Write programs that utilize global variables
- Correct errors due to variable shadowing and/or masking

# Model 1 A C++ Program

```
1  double balance = 0;
2
3  char getChoice() {
4    char userChoice;
5    cout << "Balance: \$" << balance << endl;
6    cout << "(d)eposit, (w)ithdraw, (e)xit: ";
7    cin >> userChoice;
8    return userChoice;
9  }
10
11 void deposit(double amount) {
12   balance += amount;
13 }
14
15 bool withdraw(double amount) {
16   bool success = false;
17   if(amount < balance) {
18     balance -= amount;
19     success = true;
20   }
21   return success;
22 }
23
24
```

```
1  int main() {
2    char choice;
3    double value;
4    cout << "Balance Tracker v0.1" << endl
5         << endl << fixed << showpoint
6         << setprecision(2);
7    do {
8      choice = getChoice();
9      if(choice == 'd') {
10       cout << "Enter amount to deposit \$";
11       cin >> value;
12       deposit(value);
13     } else if(choice == 'w') {
14       cout << "Enter amount to withdraw \$";
15       cin >> value;
16       if(! withdraw(value)) {
17         cout << "Insufficient Funds!" << endl;
18       }
19     }
20   } while( choice != 'e' );
21   cout << "Goodbye!" << endl;
22 }
23
```

*Refer to Model 1 above as your team develops consensus answers to the questions below.*

## Questions (15 min) Start time:

**1.** Indicate the line number on which each of the following variables is declared.

a) The variable `amount`:

b) The variable `balance`:

c) The variable `userChoice`:

d) The variable `success`:

e) The variable `choice`:

f) The variable `value`:

**2.** Categorize each of the variable declarations as one of the following:

B: Declared inside of a function body (including `main`)

P: Declared in a function header as a parameter

G: Declared somewhere else

a) `amount` c) `choice` e) `userChoice`

b) `balance` d) `success` f) `value`

**3**. One quick-and-dirty way to debug a program is to insert `cout` commands to check on the value of a variable. Suppose we inserted the given `cout` commands above the indicated line numbers (L - Left, R - Right). Place a check mark next to those commands which would compile. You can test using the file `activity14a.cpp`.

a) `cout << amount;` above line 11R

b) `cout << amount;` above line 12L

c) `cout << balance;` above line 5L

d) `cout << balance;` above line 4R

e) `cout << choice;` above line 9R

f) `cout << choice;` above line 4L

g) `cout << success;` above line 16L

h) `cout << success;` above line 20L

i) `cout << userChoice;` above line 8R

j) `cout << userChoice;` above line 6L

**4**. Pick one of the `cout` statements above which did not compile correctly. What error did the compiler report? What does that error mean?

**5**. You should have found that exactly one of the variables could be used in a `cout` statement on all the given line numbers. Which one, and why?

**6**. A *global variable* is declared outside of any function, including the `main` program and can be accessed anywhere in the program. Why do you think the programmer made `balance` a global variable?

**7**. How could you modify the code to accomplish the same tasks if you were not allowed to use a global variable?

**8.** The functions below are part of a grade book program. Define a global Boolean variable named `debug` and rewrite the functions so that when `debug==true` the function prints out the function name when it first starts executing and prints out its return values before it returns.

```
1  double minGrade(vector<double> myGrades) {
2    double tmpMin = myGrades.at(0);
3    for (int i = 1; i < myGrades.size(); i++) {
4      if (myGrades.at(i) < tmpMin) {
5        tmpMin = myGrades.at(i);
6      }
7    }
8    return tmpMin;
9  }
10
11 double avgGrade(vector<double> myGrades) {
12   double sum = 0;
13   for (int i = 0; i < myGrades.size(); i++) {
14     sum += myGrades.at(i);
15   }
16   return sum / myGrades.size();
17 }
18
```

# Model 2 A C++ Compiler Error

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5    int varOne = 5;
6    if(true) {
7      int varTwo = 7;
8      cout << varOne << endl;
9    }
10   cout << varTwo << endl;
11 }
12
```

**Compiler Error:**

```
1  model2.cpp: In function int main():
2  model2.cpp:10:11: error: varTwo was not declared in this scope
3     10 |    cout << varTwo << endl;
4        |           ^~~~~~
5
```

*Refer to Model 2 above as your team develops consensus answers to the questions below.*

## Questions (15 min) Start time:

**9**. In your own words, describe what the compiler error shown is all about.

**10**. On what lines are the following variables defined in the code above?

a) The variable `varOne`

b) The variable `varTwo`

**11**. The error message above is triggered by line 10, which comes after the variable `varTwo` has been declared. Why do you think the variable `varTwo` is not available on line 10?

**12**. The *scope* of a variable is the area of the program in which the variable is valid. Variables that have a limited scope are called *local variables* (unlike global variables, which can be accessed anywhere). In the code in this model:

- The scope of the variable `varOne` is from line 5 to line 11.
- The scope of the variable `varTwo` is from line 7 to line 9.

Based on this information, describe how to find the scope of a local variable.

**13.** The program below defines six different variables different variables. Answer the following questions about this code, which can be found in `activity14b.cpp`.

```
1  int a = 0;
2
3  void myFunc(int b = 4) {
4    int c = 5;
5    cout << "Value: " << /* OUTPUT 1 */;
6  }
7
8  int main() {
9    int d = 5;
10   if (d > 2) {
11     int e = 3;
12     if (e < 10) {
13       int f = 7;
14       cout << "Value: " << /* OUTPUT 2 */;
15     }
16     cout << "Value: " << /* OUTPUT 3 */;
17   }
18   cout << "Value: " << /* OUTPUT 4 */;
19 }
20
```

a) What is the scope of each variable?

i. `int a` – Lines to

ii. `int b` – Lines to

iii. `int c` – Lines to

iv. `int d` – Lines to

v. `int e` – Lines to

vi. `int f` – Lines to

b) Identify all global variables.

c) What variables could be used in place of each `/* OUTPUT */` comment without causing a compiler error? List all variables that would work.

i. `/* OUTPUT 1 */`

ii. `/* OUTPUT 2 */`

iii. `/* OUTPUT 3 */`

iv. `/* OUTPUT 4 */`

**14.** Why is it important to be aware of the scope of a variable?

## Model 3 A C++ Program

```
1  string str = "Global Scope";
2
3  void myFunc() {
4    cout << str << endl;
5    string str = "myFunc Scope";
6    cout << str << endl;
7  }
8
9  int main() {
10   string str = "Main Scope";
11   if (true) {
12     cout << str << endl;
13     string str = "If Scope";
14     myFunc();
15     cout << str << endl;
16   }
17   cout << str << endl;
18 }
19
```

**Output:**

```
1  Main Scope
2  Global Scope
3  myFunc Scope
4  If Scope
5  Main Scope
6
```

*Refer to Model 3 above as your team develops consensus answers to the questions below.*

### Questions (20 min) Start time:

**15.** In the program above, the `str` identifier is used for several different variables. Determine the line number of the `cout << str << endl;` command that produces each line of output. The code can be found in `activity14c.cpp`.

a) The first `Main Scope`

b) `Global Scope`

c) `myFunc Scope`

d) `If Scope`

e) The second `Main Scope`

**16.** When a variable declared within one scope blocks access to a variable of the same name declared in an outer scope (that contains the first), we say that the outer variable has been *shadowed* or that the name of the variable has been *masked*. Give two instances of variable shadowing in the code above.

**17.** Say we changed the function header on line 3 to `void myFunc(string str)` and the function call on line 14 to `myFunc("Parameter Scope")`. How would the output change?