

Name:
Sec:

09/06/2005

Lab 1.1 Logging into Educator and Creating Folders to Download Programs and Labs

This semester we will be using the online Educator system at TAMU-Commerce in order to upload your programming and lab coding assignments.

Objectives

In the first part of this lab you will learn how to log into your Educator account and set up your password. Also, you will become familiar with student folders and will create folders to hold your code from this lab and for your first programming assignment.

1. Go to the web page: <https://online.tamu-commerce.edu>
2. Sign onto your account. If you have used Educator before, you should already know your userid and have reset your password.
 - a. If you are a first time user, your userid will be composed of the first initial of your first name, followed by your full last name, followed by the last 3 digits of your student id number. For example, if I am a student whose name is John Smith and my student id number is 46193283, then my userid for Educator would be jsmith283.
 - b. Passwords for new users are initially the same as your userid, and you will be asked to change them the first time you log on.
3. Make sure you are in course Programming Fundamentals II (CSCI152001 Fall 2005)
4. Once in the online course, you will see various options on the left hand side. Click on the "Course Materials" menu item and then enter "My Folders". In "My Folders" there should be a folder with your name on it (for example John Smith Folder), enter the folder with your name.
5. When you are in a particular folder, you can do various options like "Upload to this folder", "Modify folder", "New subfolder" and "Download folder". Use the New subfolder option to create a new sub folder for this lab. Give the same name and label to the folder, call both of them Lab-1.

6. Once you have successfully created a new subfolder named Lab-1, also create a subfolder to hold your first programming assignment. Use Prog-1 for the name and label of the new subfolder.
 7. For future labs and programming assignments, you **MUST** repeat these steps to create a new subfolder (with names such as Lab-2, Prog-2, Lab-3, Prog-3, etc) in order to hold your uploaded assignments.
 8. During this lab you will be asked to write some code and upload them to your Lab-1 subfolder. Question: How will you upload a file? How will you make sure it gets uploaded into the Lab-1 subfolder as opposed to some different folder?
-

Lab 1.2 Adding User-Defined Value-Returning Functions

Because C++ does not provide every function you might ever need, you can write your own functions, called user-defined functions. User-defined functions can be value-returning functions (those that have a data type) or void functions (those that do not have a data type).

You use, or call, a value-returning function in an expression. The expression can be part of an assignment statement or an output statement. A function call in a program causes the body of the called function to execute.

A value-returning function returns a value via the return statement. The data type of the value must match the function type.

Functions in a C++ program can appear in any order. When determining the best sequence for including user-defined functions in a program, remember the rule that you must declare an identifier before you can use it. The compiler compiles the program sequentially from beginning to end.

When using some functions, such as main, which you customarily place before other user-defined functions, you can place a function prototype before the function definition to work around the problem of undeclared identifiers. When you use a function prototype, you do not have to specify the variable name in the parameter list, though you must specify the data type of each parameter. Because function prototypes appear before any function definition, the compiler translates the prototypes first so it can correctly translate a function call.

Objectives

In this lab, you include the prototype, call, and the function header of user-defined void functions, implement the missing code, and then run and test the code.

After completing this lab, you will be able to:

Use user-defined value-returning functions with and without parameters.

Adding User-Defined Value-Returning Functions

In the following exercises, you complete and test a C++ program that includes value-returning user-defined functions. The code given below may be found on the class web page at this link:

<http://faculty.tamu-commerce.edu/dharter/tamu/classes/2005fall/csci152/labs/Age.cpp>

- 1.2.a. Examine the following C++ program that calls two value user-defined functions. Provide the missing code by replacing the bold text with C++ statements, including the correct prototypes and calls to the functions. The program asks the user for today's year and month, and his or her year and month of birth. The program calculates and displays the user's age.

```

// Enter your name as a comment for program identification
// Program assignment Age.cpp
// Enter your class section, and time

/* The program Age.cpp prompts the user to input today's date and
   his/her birth date, then calculates and displays the user's
   age. The program utilizes user-defined functions.*/
/* Dates are entered in integer format for month and year. */
/* An age, described in months and years is displayed. */

//header files
/* use the correct preprocessor directives for input/output */
#include <iostream>
#include <string>

using namespace std;

// user-defined function prototypes
/* the function getValue returns an integer value
   entered by the user in response to the prompt
   in the string message */

/* the function getLetter returns a character value
   entered by the user in response to the prompt
   in the string message */

int main()
{
    // declare variables
    /* declare integer variables thisYear, thisMonth,
       year, month, and ageMonth and ageYear. declare a character
       variable again to indicate if the process is
       to continue. */
    int thisYear, thisMonth, year, month, ageYear, ageMonth;
    char again = 'y';

    // display program instructions
    cout << "This program asks you to enter today's year in 4 digits,\n"
         << "and today's month number in 2 digits.\n\n"
         << "Then it asks you to enter your birth year in 4 digits\n"
         << "and your birth month in 2 digits.\n\n"
         << "The program will calculate and display your age\n"
         << "in years and months.\n";

    // assign to thisYear the value returned from a call to getValue

    // assign to thisMonth the value returned from a call to getValue

    // loop until the user indicates to end
    while (again == 'y')
    {

        // assign to year the value returned from a call to getValue

        // assign to month the value returned from a call to getValue

        // assign age the value of thisYear - year
        ageYear = thisYear - year;
        ageMonth = thisMonth - month;

        /* test to see if thisMonth is less than month if true subtract 1
           from age and add 12 to month */
        if (thisMonth < month)
        {
            ageYear--;
            ageMonth += 12;
        }
    }
}

```

```

    }

    // display the age in years and months
    cout << "\nYou are " << ageYear << " years and " << ageMonth
        << "months old.\n"

    // assign to 'again' the value returned from getLetter

    // convert the value of again to lowercase
    again = tolower(again);

}

return 0;
}

/* the function getValue returns an integer value
   entered by the user in response to the prompt
   in the string message */
int getValue(string message)
{
    // declare variables
    // declare an integer value to enter a value
    int value;

    // prompt the user with the message passed to the function
    cout << message;

    // read in the value entered
    cin >> value;

    // return the value
    return value;
}

/* the function getLetter returns a character value
   entered by the user in response to the prompt
   in the string message */
char getLetter(string message)
{
    // declare variables
    // declare a character value to enter a value
    char letter;

    // prompt the user with the message passed to the function
    cout << message;

    // read in the value entered
    cin >> letter;

    // return the value
    return letter;
}

```

- 1.2.b. Enter, compile, link, and execute your version of **Age.cpp**. Once you have the program working to your satisfaction, save **Age.cpp** in the Lab-01 subfolder you created in your Educator account..

The following is a copy of the screen results that might appear after running your program, depending on the data entered. The input entered by the user is shown in bold.

This program asks you to enter today's year in 4 digits and today's month number in 2 digits.

Then it asks you to enter your birth year in 4 digits, and your birth month in 2 digits.

The program will calculate and display your age in years and months.

Enter today's 4-digit year: **2003**
Enter today's month number: **07**

Enter the 4-digit year of your birth: **1981**
Enter the month number of your birth: **02**

You are 22 years and 5 months old.

Do you want to enter more data? y/n **y**

Enter the 4-digit year of your birth: **1984**
Enter the month number of your birth: **12**

You are 18 years and 7 months old.

Do you want to enter more data? y/n **y**

Enter the 4-digit year of your birth: **1986**
Enter the month number of your birth: **12**

You are 16 years and 7 months old.

Do you want to enter more data? y/n **n**

Lab 1.3 Using User-Defined Value-Returning Functions

One method of designing a C++ program is to separate the design into design modules. Each module designates a different step of the program. This allows you to focus on one part of the program at a time. The process of constructing, debugging, and perfecting part of a program one module at a time is called stepwise refinement.

Objectives

In this lab, you design and write a program using design modules.

After completing this lab, you will be able to:

- ▶ Design and write a program with user-defined value-returning functions with and without formal parameters.

Using User-Defined Value-Returning Functions

In the following exercises, you design and write a program that includes user-defined value-returning functions for calculating leap years.

- 1.3.a. *Critical Thinking Exercise:* Create the design for a program that prompts the user to enter a year and then determines whether it is a leap year. Your program should have a loop and continue while the user enters the character 'y'.

Write three methods according to the following descriptions:

- `getYear` has no formal parameters, asks the user to enter a year, and returns an integer value that is assigned to the integer variable `year`.
- `isLeap` has an integer formal parameter, `year`, determines whether the year is a leap year, and returns the Boolean value `true` if the year is a leap year and `false` if it is not. A year is a leap year if it is divisible by 4, but is not divisible by 100 except when divisible by 400. (The year 2000 was a leap year.)
- `moreData` has no formal parameters and returns a `char` value.

Write your design for the main function with a loop for entering input, and the 3 user-defined functions in the following space. Your design should be a list of C++ comments without any code.

1.3.b. Write a C++ program based on the design you created in Exercise 1a and name it **LeapYear.cpp**. You will need to use your chosen compiler to create a new file. You might find it useful to add and debug each of your functions one at a time.

1.3.c. Enter, compile, link, and execute **LeapYear.cpp**. When you are satisfied with your program, save **LeapYear.cpp** in your Lab-01 subfolder in Educator..

The following is a copy of the screen results that might appear after running your program, depending on the data entered. The input entered by the user is shown in bold.

```
This program asks you to enter a year in 4 digits.
The output shows if the year is a leap year
Enter a year: 1492
1492 is a leap year.
```

```
Do you want to enter more data? y/n y
```

```
Enter a year: 2000
2000 is a leap year.
```

```
Do you want to enter more data? y/n y
```

```
Enter a year: 2005
2005 is not a leap year.
```

```
Do you want to enter more data? y/n y
```

```
Enter a year: 1800
1800 is not a leap year.
```

```
Do you want to enter more data? y/n n
```

Lab 1 Finished

You have now completed Lab 1. Turn in the previous pages with your written pseudo-code to the instructor and make sure you have uploaded your **Age.cpp** and **LeapYear.cpp** programs to your Educator account in your Lab-1 subfolder. Ask the instructor to examine your lab, which will be given either an acceptable or unacceptable evaluation. If unacceptable, you will be told of the problem areas and you may fix them and resubmit.

Attached at the end of this lab is a description of your first programming assignment, which will be due next Thursday Sept 15. After completing the first lab, now is a good time to begin thinking about how you will implement the first programming assignment. As in the **LeapYear.cpp** main function, you need to keep asking the user for input until they indicate they are done, so your main function for the first programming assignment should be similar. Think about what functions you should create to complete the assignment. The assignment description has some suggestions on functions you might want to create in order to do the program. You might want to design the functions using comments/pseudo-code below, as you did for the lab today, before actually beginning to write the program.