Exercise VII – User-defined Functions

Larry Caretto

Computer Science 106

Computing in Engineering and Science

March 21, 2006

California State University Northridge

Outline

- Exercise seven goals
- Outline tasks for exercise seven
- Provide details for task one
- Introduce task two
  - Will discuss further on Thursday
- Exercise seven is linked to the first programming project
  - Functions developed in this exercise will be used in project one
- Due date for exercise is April 4

California State University Northridge 2

Exercise Seven Goals

- As a result of this exercise you should be able to accomplish the following:
  - write programs with user-defined functions
    - function header (with argument list)
    - function body
    - function prototype
  - return values in function name
  - pass variables to functions by value and by reference

California State University Northridge 3

How do we write functions?

- C++ code is a collection of functions
- Each function, including main, has the same level of importance
  - Close code for each function before starting a new function

```
int main()
{        // body of main
}
int myFunction( …… )
{        // body of myFunction
}
```

California State University Northridge 4

Use of Functions

- A function is designed to operate by receiving data from a calling function and returning results to that function
- Once a function is written, use it for different applications by changing the data in the call to the function
- You should not have to rewrite parts of functions for different inputs

California State University Northridge 5

Tasks for Exercise Seven

- One – use of `getValidInt` function
  - Provides utility that allows you to do data validation for several int variables without need to replicate code
- Two – write functions from pseudocode to compute maximum days in a month, determine if the year is a leap year, and get valid input on month, day and year
  - Routines will be used in first project

California State University Northridge 6

Task One Functions

- `main` – you write this function to get input using the getValidInt function and write output
- getValidInt – copy this function from the assignment
- Remember to write the function prototype for getValidInt

7

getValidInt Function

- Designed to provide simple approach to obtaining integer input that is between specified maximum and minimum value
- Function parameters and minimum value, maximum value, and string description of parameter
  - Parameters passed to function by user
- Function returns valid input through function name

8

getValidInt

- Function prototype

```
        int getValidInt( int xMin,
            int xMax, string name );
```

- This function returns a value for the integer variable represented by the string, name, that is between xMin and xMax
- Example of use

```
int year = getValidInt( 1901, 2000,
  “year from the twentieth century” );
```

9

What `getValidInt` Does

- The function `getValidInt( int xMin, int xMax,string name )` does the following tasks
  - Prompts the user for an input variable (named in the string passed in the third parameter) within a range defined by the first and second parameters
  - Gets the input from the user
  - Tells the user if there is an error and gets new input from the user in this case
  - Returns valid input to the calling function

10

Using `getValidInt`

- The function `getValidInt` described on the previous chart is used in exercise seven and project one
- Use `#include <string>`
- Examples of `getValidInt` use

```
int month = getValidInt( 1, 12,
  “month”);
int mayDay = getValidInt( 1, 31, “day
  of the month” );
int year = getValidInt( 1901, 2000,
  “year in the 20th century” );
```

11

Using `getValidInt` II

- Examples of function use show different variables are input by the same function call
- Only the data and the variable in which the function result is returned change
- Do not revise function code

```
int month = getValidInt( 1, 12,
  “month”);
int mayDay = getValidInt( 1, 31, “day
  of the month” );
```

12

getValidInt Screen Results

- Call to getValidInt

```
int mayDay = getValidInt( 1, 31, “day
  of the month” );
```

- Screen prompt showing parameters (in colors) and user input

```
Enter a value for day of the month
  between 1 and 31: 0
Incorrect data; you entered day of the
  month = 0.  day of the month must be
  between 1 and 31. Reenter the data
  now.
Enter a value for day of the month
  between 1 and 31: 1
```

```
int getValidInt( int xMin, int xMax,
                 string name )
{
     // Function used to input integer
     // data within a stated range
     // Example function call to input a
     // value for a variable named
     // hour with range between 0 and 23:
     //   int hour =
     //     getValidInt( 0, 23, “hour” );

    int x;         // Input data value
    bool badData;  // Bad data flag
 // continued on next chart
```

```
do    // Loop until user data in range
{
  cout << "Enter a value for " << name
       << " between " << xMin << " and "
       << xMax << ": ";
  cin >> x;
  badData = x < xMin || x > xMax;
  if ( badData )
  {
    cout << "\n\nIncorrect data; you "
         << "entered " << name << " = "
         << x << "\n" << name;
  }   // continued on next chart
}
```

```
   do    // Same do as on previous chart
   {    //See statements on previous chart
      if ( badData ) // On previous chart
      {  //See first cout on previous chart
        cout << "must be between " << xMin
             << " and " << xMax
             << " Reenter the data now.\n";
      }
    }
    while ( badData );
    return x;
}   // end of function
```

Preview: Task Two Functions

- `main` – you write this function that calls the input function that you write
- `getMaxDays` – write this function to get maximum days in a month from pseudocode in assignment
- `leap` – write this function to tell if a year is a leap year from pseudocode in assignment
- `getValidInt` – existing function from task one
- `input` – write this function that is similar to main program from task one

Task One Requirements

- Submit listing of code and output that shows all of the following
  - Code rejects years that are too low (<1900) and years that are too high (>2000)
  - Code rejects months that are too low (<1) and months that are too high (>12)
  - Code rejects days that are too low (<1) and days that are too high (>31)
  - Code prints date for correct input
- Can do all this in one run of code

Exercise VII – User-defined Functions – Day 2

Larry Caretto

Computer Science 106

Computing in Engineering and Science

March 23, 2006

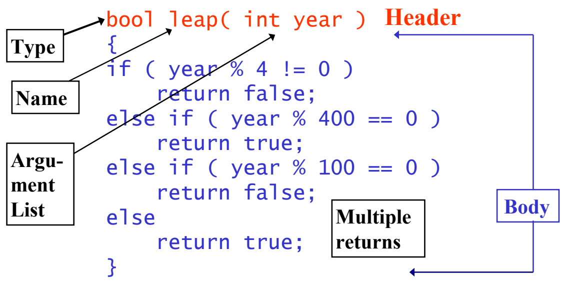
Outline

- Exercise seven goals
- Summarize lecture material on functions
- Outline tasks for exercise seven
- Provide details for some tasks
- Links between exercise seven and first programming project
- Exercise seven is due April 5 and Project one is due April 7

Exercise Seven Goals

- As a result of this exercise you should be able to accomplish the following:
  - write programs with user-defined functions
    - function header (with argument list)
    - function body
    - function prototype
  - return values in function name
  - pass variables to functions by value and by reference

Function Example



```
bool leap( int year )
{
if ( year % 4 != 0 )
      return false;
else if ( year % 400 == 0 )
      return true;
else if ( year % 100 == 0 )
      return false;
else
      return true;
}
```

Use of bool leap(int year)

```
bool leap ( year ); // prototype
int main()  // examples of use
{     cout << “Enter a year: “;
      int y; cin >> y;
      bool cond = leap( y );
      if ( leap( y ) )
      if ( leap( y ) && month == 2 )
```

Pass by Value and Reference

- Pass by value is the normal operation
  - The value of the parameter in the calling code is passed to the function
  - If the corresponding dummy parameter in the function is changed, no change is made in the parameter in the calling code
- Pass by reference is designated by ampersand (&) in header
  - Parameter passed to function can be changed

Pass by Value / Pass by Reference

Pass by Value	Pass by Reference

`// prototype`
`int x2(int x);`
`// example of use`
`int y = 5;`
`cout << x2( y )`
`     << " " << y;`
`//function`
`int x2( int x)`
`{ x = 2 * x;`
`  return x;  }`
`// output: 10 5`
	`// prototype`
	`int x2(int& x);`
	`// example of use`
	`int y = 5;`
	`cout << x2( y )`
	`     << " " << y`
	`//function`
	`int x2( int& x)`
	`{ x = 2 * x;`
	`  return x;  }`
	`// output:  10 10`

Tasks for Exercise Seven

- One – use of `getValidInt` function
  - Provides utility that allows you to do data validation for several int variables without need to replicate code
- Two – write functions from pseudocode to compute maximum days in a month, determine if the year is a leap year, and get valid input on month, day and year
  - Routines will be used in first project

Task One Functions

- `main` – you write this function to get input using the getValidInt function and write output
- getValidInt – copy this function from the assignment
- Remember to write the function prototype for getValidInt

Task Two Functions

- `main` – you write this function that calls the input function that you write
- `getMaxDays` – write this function to get maximum days in a month from pseudocode in assignment
- `leap` – write this function to tell if a year is a leap year from pseudocode in assignment
- `getValidInt` – existing function from task one
- `input` – write this function that is similar to main program from task one

getMaxDays

- Function prototype

```
int getMaxDays( int month, int year );
```

- This function returns the maximum number of days in a month for an input month (1 to 12) and year.
- Example of use

```
int mIn = 2, yIn = 2004;
int maxDays = getMaxDays( mIn, yIn );
```

leap

- Function prototype

```
     bool leap( int year );
```

- This function returns true or false if the input year is or is not a leap year.
- Example of use

```
if( leap( year ) )
{
 cout << "February has 29 days";
}
```

getValidInt

- Function prototype

```
        int getValidInt( int xMin,
             int xMax, string name );
```

- This function returns a value for the integer variable represented by the string, name, that is between xMin and xMax
- Example of use

```
int year = getValidInt( 1901, 2000,
  "year from the twentieth century" );
```

California State University Northridge 31

Input Function

- Prototype

```
void getInput( int& year,
        int& month, int& day );
```

- Use pass by reference to get values from the input function into the calling program
  - Call: `getInput( yr, mo, day );`
- Use statements like the following in the input function

```
int day = getValidInt( 1, maxDays,
              "day of the month" );
```

California State University Northridge 32

Validating Dates

- Year must be between 1900 and 2000 for exercise seven
- Month must be between 1 and 12
- Day must be between 1 and upper limit determined by getMaxDays

int maxDays = getMaxDays(month, year);
int day = getValidInt(1, maxDays, "day");
int day = getValidInt(1, getMaxDays(month, year), "day");

California State University Northridge 33

Link to First Project

- First project will use functions developed in task two to do validation of input calendar dates
- Project is to write code for astronomer's date known as Julian Day Number
- Today's class start time (12:30 pm on March 23, 2006) is 2453818.02083333 as a Julian date number
  - Fractional part is fractional part of a day starting at noon (1/48 = 0.02083333…)

California State University Northridge 34

Task Two Requirements

- Submit listing of code with functions main, getValidInt, getInput, leap, getMaxDays
- Submit output for cases shown in assignment

Set	Year	Month	Day
1	1899~2001~1990	2	0~29~28
2	2000	2	0~29
3	1990	2	29~28
4	1999	13~12	32~31
5	1999	0~6	31~30

California State University Northridge 35