Function Basics

Week 3
2017 Fall

Computer Programming for Engineers

Cassiano Campes <cassianocampes@gmail.com>
Yoohyuk Lim <dburg3065@gmail.com>
Problem 1: Let’s buy clothes
Problem 1: Let’s buy clothes (1/4)

- Write a program that asks users the following information:
  - Height, weight, and age (use proper variable types)
- These information are used to compute clothing size
- Use functions for each calculation
Problem 1: Let’s buy clothes (2/4)

- **Hat size** is equal to the weight (in **pounds**), divided by height (in **inches**) and thus all multiplied by 2.9

For conversions use these as a reference:

- 1 cm → 0.393701 in
- 1 kg → 2.20462 lbs
Problem 1: Let’s buy clothes (3/4)

- **Jacket size (in):** height times weight divided by 288
  - Then adjusted by adding one-eighth of an inch for each 10 years over age 30

Note that the adjustment only takes place after a full 10 years. So, there is no adjustment for ages 30 through 39, but one eighth of an inch is added for age 40 and so on.
Problem 1: Let’s buy clothes (4/4)

- **Waist (in):** weight divided by 5.7 and then adjusted by adding one-tenth of an inch for each 2 years over 28

Note that the adjustment only takes place after a full 2 years. So, there is no adjustment for age 29, but one-tenth of an inch is added for age 30.
Problem 1: Hints

- You probably need to create these functions:
  - Function `getHatSize(...);`
  - Function `getJacketSize(...);`
  - Function `getWaistSize(...);`
  - Function `mainMenu(...);`

- Don’t forget to use `static_cast<>` when needed
Problem 2: Let’s win the lottery price!
Problem 2: Let’s win the lottery price!

- A typical 6/49 game, each player chooses six non-duplicate numbers from a range of 1-49
- If the six numbers on a ticket match the numbers drawn by the lottery, the ticket holder is a jackpot winner - regardless of the order of the numbers

The math behind this is the **Combination** formula
Problem 2: Let’s win the lottery price!

- The Combination formula is:
  - \( n \) = the number of possible values to choose
  - \( k \) = the number of selected values

\[
C(n, k) = \frac{n!}{(n - k)!k!}
\]

- In Combination formula, does not matter the ordering
Problem 2: Hints

- Would be better to create first a function
  - `calcFactorial(...)` to calculate a single number
- Use this function to calculate the combination formula

Caution: If you use big numbers, stack overflow happens
Answer:
Problem 1
Problem 1: Solution

```cpp
#include <iostream>
#include <math.h>
using namespace std;

float convertCmToIn(int cm)
{
    /* We need to cast the integer (cm) to return a float */
    return (static_cast<float>(cm) * 0.393701);
}

float convertKgToLbs(float weight)
{
    /* Here we don't need to cast because all variables are float*/
    return (weight * 2.20462);
}

float getHatSize(int height, float weight) {
    /* Here we need to use the convert functions
    * [height] is in centimeter -> converts to inches
    * [weight] is in kilograms -> converts to pounds
    * */
    return (convertKgToLbs(weight) / convertCmToIn(height) * 2.9);
}
```
Problem 1: Solution

```c
25. float getJacketSize(int height, int age, float weight) {
26.     float result;
27.     float adjust = 0;
28.     
29.     /* First we check if the age is above the adjustment condition
30.     * If so, we use modulus operator to get the remainder of the division
31.     * and thus we can determine if we need to subtract or not
32.     * the age to identify how many times we must add the adjustment.
33.     * Otherwise, the adjustment is left as 1, and does not affect the
34.     * multiplication in the return statement of the function.
35.     **/
36.     if (age >= 30) {
37.         if ((age % 10) != 0)
38.             age = age - (age % 10);
39.         adjust = (1.0/8)*((age - 30) / 10);
40.     }
41.     
42.     /* We convert the values and multiply accordingly to what is requested */
43.     result = ((convertCmToIn(height) * convertKgToLbs(weight)) / 288);
44.     
45.     /* Returning the result from the function. Notice that we had previously
46.     * calculated the adjust variable, that can be >= 1 */
47.     return (result + static_cast<float>(adjust));
48. }
```
Problem 1: Solution

```c
float getWaistSize(int age, float weight) {
    float result;
    float adjust = 0;

    /* Same as the above mentioned about checking the age for the adjustment */
    if (age >= 28)
    {
        if ((age % 2) != 0)
            age = age - (age % 2);
        adjust = (1.0/10)*((age - 28) /2);
    }

    /* We convert the value and multiply accordingly to what is requested */
    result = (convertKgToLbs(weight) / 5.7);

    /* Returning the result from the function. Notice that we had previously
    * calculated the adjust variable, that can be >= 1 */
    return (result + static_cast<float>(adjust));
}

int Menu() {
    /* Store inputs from the user */
    int height;
    int age;
    float weight;
```
Problem 1: Solution

```cpp
/* Stores the output to be given to users */
float hat_size;
float jacket_size;
float waist_size;

/* Let's show something to the user */
cout << "Input your [height] in cm, [weight] in kg, and [age] in years:" << endl;

/* Get the inputs from the user, notice that we concatenate the variables,
* thus, the inputs must be separated by spaces */
cin >> height >> weight >> age;

cout << "Hat size: " << getHatSize(height, weight) << endl;
cout << "Jacket size: " << getJacketSize(height, age, weight) << endl;
cout << "Waist size: " << getWaistSize(age, weight) << endl;
}

int main() {

/* Set the format for the floating point variables */
 cout.setf(ios::fixed);
 cout.setf(ios::showpoint);
 cout.precision(2);
```
Problem 1: Solution

/* This is an auxiliary function called to simplify the main() function
   * Menu() is in charge of calling all the other functions accordingly */

Menu();

return 0;

}
Answer:
Problem 2
Problem 2: Solution

```cpp
#include <iostream>
#include <stdio.h>
#include <math.h>
using namespace std;

/* This is the basic function that is used to calculate factorial
   * As you can notice, the function calls itself by decrementing
   * its iteration index 'n'.
   * Every time the function calls itself, it is decremented and
   * once the index reach the value of 1, the function
   * returns.
   * Let's suppose we called calcFactorial(4), the behavior is
   * as follows:
   * calcFactorial(4)
   * \--> calcFactorial(3)
   * \--> calcFactorial(2)
   * \--> calcFactorial(1)
   * At this point 1 is reached, then, the functions
   * starts to return the value */

long calcFactorial(long n) {
    long tmp = 0;
    if (n <= 1) {
        return 1;
    }
}
```
Problem 2: Solution

```cpp
return (n * calcFactorial((n - 1)));
}

/* This function calculates the combination formula
 * We use the calcFactorial primitive function to get
 * the calculation done */
long calcCombination(long n, long k) {
    return (calcFactorial(n) / (calcFactorial(k) * calcFactorial((n-k))));
}

/*
 * This is the main function where we get the input from the user
 * about the combination to be calculated */
int main() {
    long n, k;
    cout << "Enter the [n] and the [k] combination values: " << endl;
    cin >> n >> k;
    cout << calcCombination(n, k) << endl;
    return 0;
}
```