CS1010E: Programming Methodology

Tutorial 01: Identifiers, Types, and Operations

23 Jan 2017 - 27 Jan 2017

1. Discussion Questions

(a) [Identifiers] Which of the following identifier(s) is/are valid?

A. `A_Long_Identifier`	E. `year#2`	I. `valid identifier`
B. `module code`	F. `1010E`	J. `_invalid`
C. `SG$`	G. `CS1010E`	K. `x`
D. `int_val`	H. `$USD` (*this is* ***invalid***)	L. `_valid?_`

(b) [Types] Given the following program fragment, what are the values and types of ALL the declared variables ? (**note**: *give* **`real`** *number up to 6 decimal places*)

i. `int i = 10;` i. ____________

ii. `int j = 12.3;` ii. ____________

iii. `int i = 123, j = 456.789;` iii. ____________

iv. `double f = 1010;` iv. ____________

(c) [Operation Precedence] What is the final value of `ans` in the code fragment below?

i.
```
int a = 1, b = 2, c = 3, d = 4;
int ans = a + b * c + d;
```
i. ____________

ii.
```
int a = 1, b = 2, c = 3, d = 4;
int ans = a + b - c + d;
```
ii. ____________

iii.
```
int a = 1, b = 2, ans = a++ + b++;
    ans = ans + a + b;
```
iii. ____________

iv.
```
int a = 1, b = 2, ans = ++a + ++b;
ans = ans + a + b;
```
iv. ____________

2. Program Analysis

(a) [Division and Modulo] What is the final value of `ans` in the code fragment below? (**note**: *give* **`real`** *number up to 3 decimal places*)

i.
```
int a = 5, b = 2;
double ans = a / b * 1.0;
```
i. `ans = 2.000 (integer division)`

ii.
```
int a = 5, b = 2;
double ans = a * 1.0 / b;
```
ii. `ans = 2.500 (implicit type conversion)`

iii.
```
int a = 5, b = 2;
int ans = a * 1.0 / b;
```

iv.
```
int a = 5, b = 2;
int ans = a / b * 1.0;
```
iv. `ans = 2 (truncation)`

iii. `ans = 2 (truncation)`

v.
```
int matric = 0040607,
    ans    =  matric%10
            + (matric/100)%10
            + (matric/10000)%10;
```
v. `ans = 17 (modulo and digit extraction)`

(b) [Limit of Values] What is the final value of `ans` in the code fragment below?

i. `short a = 32767, ans = a + 1;`

i. `ans = -32768 (overflow short` $2^{15}-1$`)`

ii. `unsigned short b = 65535, ans = b + 1;`

ii. `ans = 0 (overflow unsigned` $2^{16}-1$`)`

iii.
```
double c   = 9007199254740992,
       ans = c + 1;
```
iii. `ans = 9007199254740992 (limit of double)`

3. Designing a Solution

(a) [Computation] Consider a bank account with an initial value given in a variable `balance` and the annual interest rate of the bank (*as a percent value*) given in a variable `rate`. Assuming that the the calculation of the *annual compounded interest* for 3 years can be exemplified by the calculation below involving the initial balance of `1500` and the annual interest rate of `2.5%`. The calculations below are rounded to 2 decimal places.

```
0-th year balance  = $1500.00 (initial balance)
1-st year interest = $1500.00 × 0.025  = $37.50
1-st year balance  = $1500.00 + 37.50  = $1537.50
2-nd year interest = $1503.75 × 0.025  = $38.44
2-nd year balance  = $1503.75 + 38.44  = $1575.94
3-rd year interest = $1507.51 × 0.025  = $39.40
3-rd year balance  = $1507.51 + 39.40  = $1615.34
```

Write a program to compute the balance at the end of the 3^{rd} year given any *positive* value of `balance` and `rate`. Write your program below:

```
int main() {
  double balance, rate;
   /* Calculate Final Balance Here */
   balance += balance * rate / 100.0; // interest = (balance * rate / 100.0)
   balance += balance * rate / 100.0; // balance  = balance + interest
   balance += balance * rate / 100.0; // **repeat 3x**
  printf("%.2f\n", balance);
  return 0;
}
```

(b) [Computation] We simplify the question regarding bank account above to compute not the amount after *compounded interest* but the amount after *simple interest*. The calculations are now changed to be:

```
0-th year balance  = $1500.00 (initial balance)
interest           = $1500.00 × 0.025  = $37.50
1-st year balance  = $1500.00 + 37.50  = $1537.50
2-nd year balance  = $1537.50 + 37.50  = $1575.00
3-rd year balance  = $1575.00 + 37.50  = $1612.50
```

However, besides the initial balance and the interest rate (*now called simple interest rate*), we will also include the number of *months* the account have been active. If there are no additional balance added to the account, the interest rate remains constant throughout the year, and the interest is only added at the end of the year (*ignoring partial year*), calculate the balance *at the end of the month*. Write your program below:

```
int main() {
  double balance, rate; int month;
   /* Calculate Final Balance Here */
   int year = month/12;                        // ***Alternative Method***
   double interest = balance * rate / 100.0; // interest = 1.0 + (year * rate / 100.0);
   balance += interest * year;                 // balance = balance * interest;

  printf("%.2f\n", balance);
  return 0;
}
```

4. Challenge

(a) [Design] Lecture Notes 1 includes the algorithm and code for the greatest common divisor (GCD) of two **integer**. Using the *4-steps methodology* described in the Lecture Notes 1, solve the problem of finding the least common multiple (LCM).

i. **Understanding the Problem:**
Inputs are two numbers `x` and `y`, find `LCM(x,y)`.
Assume that `x` and `y` are positive.
More precisely, assume $x \geq y > 0$.

ii. **Devising a Plan:**
This problem is very similar to GCD.
In fact, if the solution to GCD of two numbers `x` and `y` is `GCD(x,y)`,
the value of `LCM(x,y)` is simply $(x \times y)/GCD(x,y)$.
Thus, we can break this problem into **two (2)** smaller steps:
1. Finding `GCD(x,y)`, and
2. Compute $(x \times y)/GCD(x,y)$

iii. **Carrying Out the Plan:**
The implementation is:
1. Finding `GCD(x,y)`: use the algorithm in Lecture Notes 1
2. Compute `(x` $\times$ `y)/GCD(x,y)`: compute using the formula `(x*y)/GCD(x,y)`

iv. **Looking Back:**
Test the result using different values until confident.