Basics of Programming

L03: Program Design

March 2020

Dr. Ram P Rustagi
Professor, CSE Dept
KRP, KSGI
rprustagi@ksit.edu.in

Review: Last Lectures

- Wrote simple programs (graphics, polygons)
- Basic arithmetics
- Need for variables
- Need for loops
- Need for functions

How to Write Programs

- Ensure program works correctly for all valid inputs
  - It should reject invalid/illegal inputs
- Program should never crash
  - it should do a graceful handling
- Writing a program requires some planning
  - A logical thinking and algorithmic analysis,
- Expectation when writing a program
  - Correct
  - Maintainable
  - Elegant
  - Meets performance objectives

Program Development

- Understand requirements and objectives
- Write specifications
- Identify and construct the test cases
- Analyze and think how would you solve the problem with pencil and paper.
  - A must to write correct programs
- Write down your ideas formally and make a plan
- Write (code) the program. Preferably use IDE
- Perform mental check if the program follows your plan. Are there any mistakes in program writing
- Run the test cases you have planned.
- Debug the challenges. Use debuggers.
  - Avoid print statements for debugging

Programming Problem

- Computation of `e`:
  `e=1/0! + 1/1! + 1/2! + …+ 1/n!`
- Write the program with following variations:
  – Take `n` as an input and computes `e`.
- 2nd variation of this program:
  – Take δ as input and stop when incremental change becomes less than δ.

Specification

- Specification
  - What is input?
  - What is output?
  - When will you consider output as correct
- Real life programming problems
  - There may be ambiguity and confusion
  - Write down what is given
  - What is needed precisely
  - Write down your assumptions
  - Identify conditions (inputs) when your program will not work

Specification for computing `e`

- Input
  - An integer `n≥0`
- Output
  - Sum: `1/0! + 1/1! + 1/2! + …+ 1/n!`
- Notes:
  - Specified that input n is a positive integer.
    - Can't be a negative number
    - Can't be real number.
    - Can we mistakenly assume something.
      - How many terms to be added
        - » `n` or `n+1`?
      - How many additions:
        - » `n` (and not `n+1`)

Test cases for computing `e`

- Write initial few computation to help better understanding
- Computation answer for some values of n
  - `n=0, ans=1`
  - `n=1, ans=2`
  - `n=2, ans=2.5`
  - `n=3, ans=??`
    - it is not `2.5+1/3,`
    - it is `2.5+1/6`

Algorithm for computing `e`

- Pen and pencil approach
  - calculate `1/0!`
  - calculate `1/1!` and add to previous value
  - calculate `1/2!` and add to to previous sum
  - calculate `1/3!` and add to previous sum
- How should you calculate each term
  - Independently, or
  - Make use of earlier tem
- What is the formula for computing k^{th} term
  - k^{th} term = $(k-1)^{th}$ term $/k$
  - `1/k! = (1/(k-1)!)/k`
- Now think of the program to write

Consolidations of thoughts

- Computing `e`
  - Program must perform `n` additions
    - Have a loop that iterates `n` times.
  - In the k^{th} iteration, compute k^{th} term
    - Add to previous sum
  - To compute k^{th} term, we need $(k-1)^{th}$ term
    - need to know the value of `k`
  - How many variables we need
    - sum
    - last term
  - For loop iteration use the iterator i.e. `i`

Program Sketch

```
main {
    int n;
    get n; //either command line arguement, or read
    double sum=0.0, term=0.0;
    int i;// loop variable
    for i=0 to n {
      // is it as per our thoughts?
      term = term/i
      sum = sum + term
    }
    print sum
}
```

Testing and Debugging

- Run for different values of input n
- Use IDE to debug
- Use meaningful comments on what the program is doing.
- Get your code review done by your colleagues.
  - Can s/he understand it without you explaining it.
- Do not use any hard coding of values.
  - Use parameters, variables etc.

Programming Exercises

- A:Compute the following for n terms

$$1.\ e^x = \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$$

$$2.\ \frac{2}{\pi} = \frac{\sqrt{2}}{2} \cdot \frac{\sqrt{2+\sqrt{2}}}{2} \cdot \frac{\sqrt{2+\sqrt{2+\sqrt{2}}}}{2} \cdots$$

- B: Compute `D(r)`, which is the number of ways in which numbers `1` thru `r` can be arranged in a sequence such that `i` is never in the `i`th position for all `i`.

$$D(r) = \sum_{k=0}^{r} (-1)^k \frac{r!}{k!}$$

Exercise Review

- Print first N prime numbers.
- Approach
  - Start with count=0
  - Increment by 1 when next prime number is printed
  - Stop when count becomes N
  - Start with first integer (=2)
  - Check for primality, if yes
    - print the number, increase count
  - Proceed with next number
- Algo:
  - one function to check for primality e.g. prime(x)
  - one loop to keep track of count

Exercise Review

```
import math
def prime(x):
 for i in range(2,int(math.sqrt(x))+1):
   if x % i == 0:
     return False
 return True
#main code
N=10 # count of prime numbers
cnt=0;
num = 2 # start from 2
while cnt < N:
  if prime(num):
    cnt = cnt + 1
    print(num)
  num = num +1
#————————————–
```

Programming Exercises

- C: Write a program that implements La-Russe algorithm for multiplication of two numbers A &B.
  - The algo works as follows,
    - Divide A by 2 and multiply B by 2.
    - Repeat the above process till A becomes 1.
    - For all those combinations of A and B, whenever A is odd, add all such values of B
    - The result will be multiplication of two numbers.
    - You should be able to do it only using one extra variable other than that for A & B

Programming Exercises

- D: write a program that computes maximum and minimum of two numbers A & B without using any direct comparison operation between these two numbers. You can use comparison with 0 (Zero)
  - Hint: use absolute function of maths.

Summary

- How to write correct programs
- Consider an implementation using pencil and paper.
- Identify few test cases.
- Identify where it can go wrong
- Get your code review done.

Questions