Indian Institute of Technology Mandi

IC150: Computation for Engineers

Tutorial 2 9th April 2014

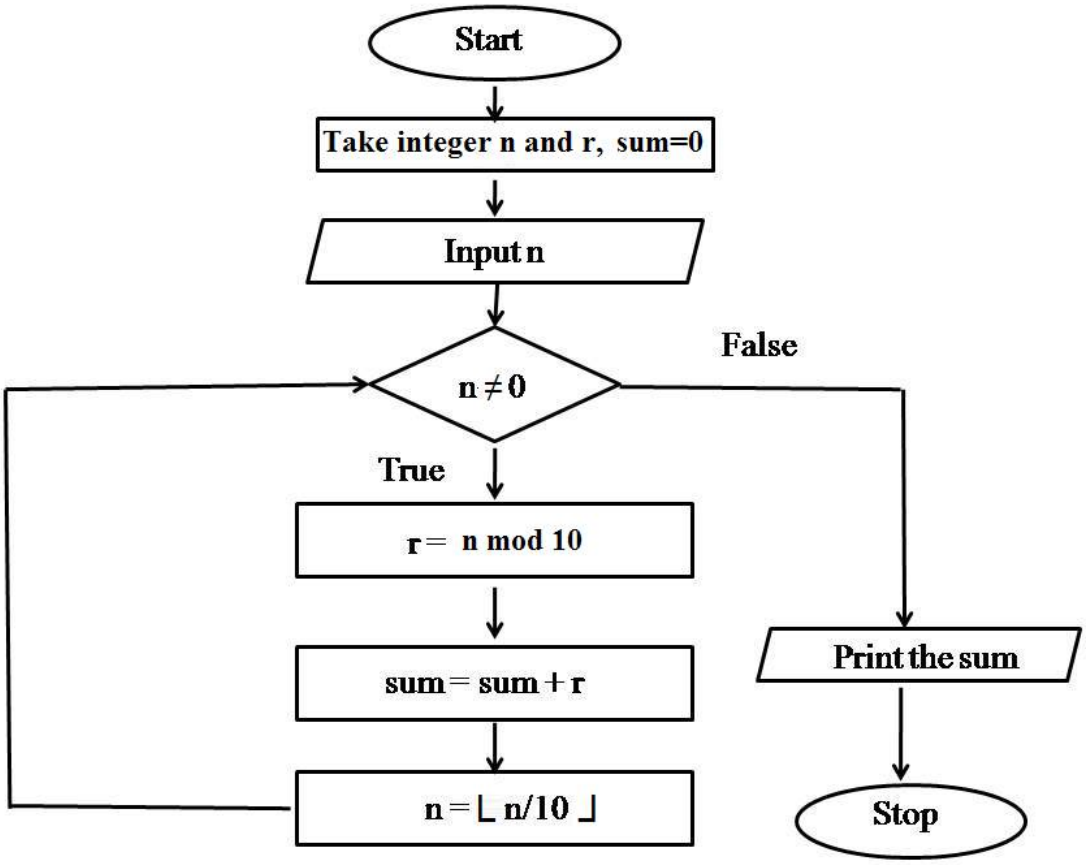
Ques:1 The coefficients of two *n*-degree polynomials are stored in the arrays *p1* and *p2*. Write pseudo-code to add the two polynomials and store the result in array *p3*. Do **not** write C code.

1. Let p1[0..n] and p2[0..n] contain the n+1 coefficients of the polynomials

2. For each i in 0 to n, do

2.1 p3[i] ← p1[i]+p2[i]

Ques:2 Draw a neat flowchart to read an integer *n*, add its digits and print the result. Eg, given *n* = 726, the output is 15.



Ques:3 A simple calculator, `scalc`, takes several integers on the command-line and computes their average. If the 1st argument is "`-t`", it prints the total instead of the average. Eg, the command

```
$ scalc 4 5 -7 12
```

prints `3.5`, and

```
$ scalc -t 4 5 -7 12
```

prints `14`. If no arguments are given, it prints an appropriate error message. Write C code for `scalc.c`.

Solution:

```
#include <stdio.h>
#include <string.h>
void PrintUsage()    // Print help info and exit
  {
    printf("Usage: scalc.exe  4 5 -7 12 \n");
    printf("       scalc.exe  -t 4 5 -7 12\n");
    printf("scalc -t requires at least 1 arg\n");
    exit(1);
  }

// Given a[] containing integers in the form of strings, return the sum of a[begin] … a[begin+cnt]
int Sum(char *a[], int begin, int cnt)
     {
          int j;
          int tot=0;
          for(j=begin; j<cnt; j++) tot += atoi(a[j]);

          return tot;
     }

void main(int argc, char *argv[])
  {
     int start = 1, numCnt;
     int total = 0;
     int totFlag = 0;     // True if -t option
     numCnt = argc-1;     // Count of numbers to add if no -t option

     if (!totFlag && numCnt <= 1)
        PrintUsage();     // error exit, need at least 1 number for average

     if (strcmp(argv[1], "-t") == 0)
       {
         totFlag = 1;
         numCnt--;        // Skip the -t argument
         start++;
       }

     if (numCnt <= 0) PrintUsage();      // error exit

     total = sum(argv, start, numCnt);

     if (totFlag)
         printf("Total = %d\n", total);
     else
       printf("Average = %f", (float) total/numCnt);
  }
```

Ques:4 Define a type `ComplexType` that can hold a complex number. Write the following functions:

`MakeComplex(x, y)` – returns a complex number <x + iy>
`AddComplex(c1, c2)` – returns the sum of the two complex numbers `c1` and `c2`
`MultComplex(c1, c2)` – returns the product of the two complex numbers `c1` and `c2`

Solution:

```
typedef struct
  {
     int real;
     int imag;
  } ComplexType;
ComplexType MakeComplex(int x, int y)
  {
     ComplexType c;
     c.real=x;
     c.imag=y;
     return c;
  }
ComplexType AddComplex(ComplexNum c1, ComplexNum c2)
  {
     ComplexType sum;
     sum.real = c1.real+c2.real;
     sum.imag = c1.imag+c2.imag;
     return sum;
  }
ComplexType MultComplex(ComplexNum c1, ComplexNum c2)
  {
     ComplexType prod;
     prod.real = c1.real*c2.real-c1.imag*c2.imag;
     prod.imag = c1.real*c2.imag+c2.real*c1.imag;
     return prod;
  }
```

Ques: 5 Write the function `strend(s, t)` which returns `1` if the string `t` occurs at the end of the string `s`, and `0` otherwise.

Solution:

```
int strend(char *s, char *t)
  {
     int ls = 0, lt = 0;
                                   /* Find lengths of strings s and t, excl. \0  */
     while (*(s+ls)) ls++;
     while (*(t+lt)) lt++;
     assert(lt <= ls);

                                   /* Compare the last lt chars of s and t */
     s = s+(ls - lt);
     while (*s)
         if (*s++ != *t++) return 0;      // Doesn't match
         else return 1;
}
```