Indian Institute of Technology Mandi

# IC150: Computation for Engineering

## Tutorial 5

1) For each of the following, state whether it is a valid C variable name. If it is not valid, explain why and correct it to make it valid.

   (a) `6thBTech`

   Invalid. First character must be an alphabet or '_'.
   Valid: BTech6th, SixBTech

   (b) `boysNgirls`

   Valid.

   (c) `ladies&gents`

   Invalid. '&' not allowed. Valid: ladiesNgents

   (d) `Num_Hostel-Rooms`

   Invalid. '-' not allowed. Valid: NumHostelRooms

2) The cutoffs used in grading a particular course are: below 50: D; 50 to 59: C ; 60 to 75: B; 75 above: A.

   (a) Given the variable `marks`, write C code using only `if ... else` to assign `'A'`, `'B'`, `'C'` or `'D'` to the variable `grade`.

```
if (marks<50)
    printf("grade='D'");
else if (marks<60)
    printf("'C'");
else if (marks<76)
    printf("'B'”);
else
    printf("'A'");
```

   (b) Rewrite the code in (a) without using any of the decision control structures `if ... else`, `switch` or `(...)?...:...`

Aliter 1: use a lookup table lut[] that stores the grade for each mark from 0..100

```
char lut[101];
int m;
                                          // Initiallise the lookup table
for (m = 0; m < 50; m++)     lut[m] = 'D';
for (m = 50; m <= 59; m++)   lut[m] = 'C';
for (m = 60; m <= 75; m++)   lut[m] = 'B';
for (m = 76; m <= 100; m++) lut[m] = 'A';

grade = lut[mark];                        // Use the lookup table
```

Aliter 2: use the fact that a boolean False or True is integer 0 or 1 respectively and that the 4 letter grades have consecutive positions in the ASCII table

```
grade = 'A' + (mark <= 75) + (mark<=59) + (mark < 50);
```

3) Write a program that reads input characters and exits when a pre-specified number of occurrences of any character is read. First decide on data structures (variables), next design the procedure to solve the problem, using step-wise refinement. Use `ch = getchar()` for input of a character.

```
#include <stdio.h>
int main()
 {
     int count[256];                    /* number of each character seen */
     int ch, i;
     int maxOccur = 3;

     for (i=0; i<256; i++) count[i] = 0; // Zero the count array

     while ((ch = getchar()) != EOF)
       {
         putchar(ch);                   // echo the input to output
         if (++count[ch] >= maxOccur)
           {
               printf("\nSeen %d occurrences of %c\n",
                          count[ch], ch);
               return(0);
           }
       }
     return(1);             // input ended before maxOccur chars
}
```

4) Given date1 = d1/m1/y and date2 = d2/m2/y (note that y is the same in both), write pseudo-code to compute the number of days in the period from date1 to date2, both inclusive.

```
dim[12]  {31,28,...,31}              // days in month 1 .. 12

if ((y is divisible by 100 and y is divisible by 400) OR
   (y is not divisible by 100 and y is divisible by 4))
   dim[2]     29
if (m1 == m2) p = d2-d1+1            // Assume d1/m1 earlier than d2/m2
else
   for m from m1 to m2 in steps of 1 do
     if(m==m1) p     p+dim[m]-d1+1 // First month, count partial
     else if (m==m2) p     p+d2     // Last month, count partial
     else p     p+dim[m]            // Middle month, count full
```

5) Design an ADT for storing and manipulating polynomials. The ADT has the following functions:

`Poly MakePoly(int n);` // Creates and returns an empty polynomial of order `n`

`int SetPolyTerm(Poly p, int i, int coeff);`
// Set the `i`th term to `coeff`. Returns 0 if `i` > order of `p`, else returns 1

`Poly AddPoly(Poly p1, Poly p2);`
// Return the sum of `p1` and `p2` in a new polynomial

`void PolyPrint(Poly p1);` // Prints the polynomial p1 in a human-readable format

(a) Design a structure to hold a polynomial and hence define the type `Poly` using `typedef`.

(b) Write a header file `poly.h` that defines the `Poly` type and the function prototypes.

(c) Write the implementation file `poly.c` containing the code for the functions `MakePoly`, `SetPolyTerm` and `AddPoly`.

(d) Write a file `testpoly.c` that contains a main function to test the polynomial ADT.

(e) Write the command(s) that you would use to compile the code, saving the executeable in the file `polytest`. Assume that the file `PrintPoly.c` contains the implementation of the function `PolyPrint()`.

```
// poly.h
#ifndef _POLY_H_
#define _POLY_H_
typedef struct _poly * Poly;  // declaration of the ADT
Poly MakePoly(int n);   // Creates and returns an empty polynomial of degree n
int  SetPolyTerm(Poly p, int i, float c); // Set the ith term to coeff.
Poly AddPoly(Poly p1, Poly p2); // Return sum of p1 and p2 in a new poly
void PolyPrint(Poly p1);  // Prints poly p1 in a human-readable format
#endif
```

// **poly.c**

```
#include<math.h>
#include<stdlib.h>
#include"poly.h"

struct _poly {
     int deg;
     float *coeff;
};

Poly MakePoly(int n)

{
                    // allocate a Poly structure
     Poly p=(Poly)malloc(sizeof(struct _poly));

     p->deg=n;

     if(n>0){

          // allocate an array of length n+1
          p->coeff=(float *)malloc(sizeof(float)*(n+1));

          int i;
          for(i=0;i<n+1;i++)
               p->coeff[i]=0.0;      // set all coeff to 0

     }

     else
          p->coeff=NULL;

     return p;
}

int SetPolyTerm(Poly p, int i, float c)

{
// Returns 0 if i > order of p, else returns 1

     if(i>p->deg)
          return 0;

     else
     {
          p->coeff[i] = c;
          return 1;
     }
}
```

```
Poly AddPoly(Poly p1, Poly p2)
     {
     int i,n;
     Poly p3;
     // Find the maximum degree of two polynomials
     n = max(p1->deg,p2->deg);
     p3= MakePoly(N);
     // Addition of two polynomial
     if(n==p1->deg){   // Add the coeff. upto lower degree polynomial
          for(i=0;i<=p2->deg;i++)
               p3->coeff[i]=p1->coeff[i]+p2->coeff[i];
          // Assign the coeff. of higher degree polynomial
          for(i=(p2->deg+1);i<=p1->deg;i++)
               p3->coeff[i]=p1->coeff[i];
     } else {
          for(i=0;i<=p1->deg;i++)
               p3->coeff[i]=p1->coeff[i]+p2->coeff[i];
          for(i=(p1->deg+1);i<=p2->deg;i++)
               p3->coeff[i]=p2->coeff[i];
     }
     return p3;
     }
```

// **testpoly.c**

```
#include<stdio.h>
#include"poly.c"
#include"PrintPoly.c"
int main(){
     Poly p1,p2,p3;
     p1=MakePoly(3);  // Create p1 with 3 degree
     // Set the coeff. of  p1
     SetPolyTerm(p1,0,2.1);
     SetPolyTerm(p1,1,1.2);
     SetPolyTerm(p1,2,4);
     SetPolyTerm(p1,3,7);
     p2=MakePoly(4);  // Create p2 with 4 degree
     // Set the coeff. of  p2
     SetPolyTerm(p2,0,2);
     SetPolyTerm(p2,1,6);
     SetPolyTerm(p2,2,1.5);
     SetPolyTerm(p2,3,5.6);
     SetPolyTerm(p2,4,3);
     // Add p1 and p2
     p3=AddPoly(p1,p2);
     printf("First Polynomial:\n");
     PolyPrint(p1);
     printf("Second Polynomial:\n");
     PolyPrint(p2);
     printf("Addition of Polynomials:\n");
     PolyPrint(p3);
}
```